Escola Politècnica Superior
de Castelldefels
UNIVERSITAT POLITÈCNICA DE CATALUNYA

TREBALL DE FI DE CARRERA

TÍTOL DEL TFC: Disseny i implementació d'un *embedded system* basat en PSoC de Cypress: tauler visualitzador de bàsquet

TITULACIÓ: Enginyeria Tècnica de Telecomunicació, especialitat Sistemes de Telecomunicació

AUTOR: Robert Pinedo López

DIRECTOR: Francesc Josep Sànchez i Robert

DATA: 22 de gener de 2010

Títol: Disseny i implementació d'un *embedded system* basat en PSoC de Cypress: tauler visualitzador de bàsquet

Autor: Robert Pinedo López

Director: Francesc Josep Sànchez i Robert

Data: 22 de gener de 2010

Resum

L'objectiu del projecte es introduir al lector en el món dels microcontroladors PSoC de Cypres a través del disseny i el muntatge d'un aparell electrònic senzill com és un marcador de bàsquet.

El projecte està dividit en tres parts, seguint uns passos lògics necessaris per aconseguir l'objectiu final.

El primer capítol és una introducció de l'aplicació a desenvolupar. Conté una recerca d'aparells semblants ja existents i una sèrie d'especificacions a complir. També s'hi proposa un diagrama de blocs.

Al segon capítol el lector pot trobar el disseny, el muntatge i les verificacions de cada bloc del diagrama. Aquesta part demostra que el muntatge final del marcador és factible.

El tercer capítol conté l'explicació del muntatge de l'aparell sencer. En ell es verifica que el marcador compleix els objectius proposats inicialment.

Title: Design and implementation of an embedded system based on Cypress PSoC: basketball electronic scoreboard

Author: Robert Pinedo López

Director: Francesc Josep Sánchez Robert

Date: January, 22th 2010

Overview

The purpose of this project is to introduce the reader to the world of Cypress PSoC microcontrollers through the design and assembly of a simple electronic device as a basketball scoreboard.

It is divided into three parts, following logical steps necessary to reach the objective.

The firts chapter is an introduction to the application to develop. In it, a research about similar devices and a list of objectives is made. Then, a bloc diagram is proposed.

In the second chapter the reader can find the design, the assembly and the test of each part of the bloc diagram. This part shows that the final assembly of the scoreboard is possible.

Then, the third chapter contains the assembly of the entire device. The scoreboard is verified and the initial objectives are accomplished.

Gràcies al Francesc Josep Sànchez per guiar-me en el projecte,
al Francis López per ajudar-me amb l'EAGLE
i al Sergio García i al Juan José Tomeo per fer-me unes bones plaques.

ÍNDEX

INTRODUCCIÓ

En aquest projecte es dissenya i es construeix un prototip de marcador electrònic de basquetbol basat en un microcontrolador PSoC de la marca Cypress. L'objectiu és introduir al lector en el món dels microcontroladors, i més concretament del PSoC, a través d'aquest exemple pràctic.

La divisió en capítols que s'ha aplicat pretén seguir cronològicament un ordre adient per dissenyar i muntar qualsevol tipus d'aparell electrònic controlat per un microcontrolador.

En el primer capítol s'introdueix l'aparell que es vol construir. Es fa una recerca d'informació sobre aparells del mateix tipus per elaborar un estudi acurat dels requeriments que ha de complir. En base a això es decideix l'estructura que tindrà i es representa mitjançant un diagrama de blocs.

El segon capítol es divideix en 4 apartats. En els dos primers s'analitza cada un dels blocs principals del diagrama. S'estudien les funcions que han de complir i s'escullen els tipus de dispositius electrònics que es necessiten. Es tenen en compte les seves possibilitats, les seves característiques, la seva disponibilitat en el mercat i la compatibilitat de cada un d'ells amb la resta de dispositius als quals es connectin. També es descriu el tipus de comunicacions digitals que utilitzen.

Al tercer apartat s'escull el microcontrolador al qual van connectats directa o indirectament tots els dispositius. Es comprova que és compatible amb ells i que és capaç de comunicar-s'hi.

En l'últim apartat es verifica que cada bloc funciona correctament. Es dissenyen, es munten i es verifiquen petits circuits de prova que emulen cada funció de l'aparell final.

En el tercer capítol es mostra el disseny final basat en les proves fetes al capítol anterior. Es dissenyen i es fabriquen dues plaques de circuit imprès on es solden els components i s'obté l'aparell final. També es dedica un apartat a analitzar els criteris ambientals que es compleixen en la fabricació de cada peça i un darrer apartat a les conclusions personals.

En el quart i últim capítol es mira cap al futur per pensar quines millores es podrien afegir al prototip que es poguessin realitzar en nous projectes. També s'hi inclou la bibliografia i els annexos. En aquests últims es pot trobar un tutorial que mostra pas a pas com s'ha utilitzat el *software* de programació del microcontrolador i el codi en llenguatge C que s'ha fet servir.

CAPÍTOL 1. Introducció a l'aplicació a desenvolupar. Diagrama de blocs general

1.1 Marcador de bàsquet

El marcador de bàsquet és un element informatiu i de control que durant un partit indica el temps de joc, el tempteig, les faltes acumulades i altres característiques de l'encontre que ajuden als dos equips, als àrbitres i al públic a situar-se en el context dels esdeveniments.

Ha de ser visible des dels diferents punts del camp de joc i dels seus voltants tant pels jugadors i equips tècnics com pel públic i els mateixos auxiliars de taula. Ha de tenir, per tant, unes dimensions, una intensitat lluminosa i un so adequats.

El marcador és controlat pels auxiliars de taula, que en bàsquet són els encarregats de controlar el temps dels períodes, el de possessió de pilota i l'acta que resumeix exhaustivament els esdeveniments clau del partit. El marcador ha de disposar d'una consola de control des de la qual sigui possible gestionar aquestes funcions.

Segons el reglament de la Federació Catalana de Basquetbol (FCBQ) un marcador ha de mostrar com a mínim (textualment):

- *"El temps de joc.*
- *La puntuació arrossegada.*
- *El número de període actual, que es puja quan ha de començar el període.*
- *El número de temps morts registrats, que es pugen quan s'inicia el temps mort enregistrat.*
- *"Pilots" lluminosos corresponents a les faltes d'equip o el número de faltes d'equip realitzades."*

També especifica que els marcadors indicaran a més a més:

- *"El nombre de faltes d'equip, d'1 a 5 (amb possibilitat d'aturar en arribar al màxim de 5).*
- *El nombre de període d'1 a 4 i "E", per un període extra.*
- *El nombre de temps morts, de 0 a 3. En el cas de que només es mostrin 2, es marcaran fins que sigui possible."*

Altres característiques extretes del mateix document són:

- *"Els marcadors incorporaran un rellotge digital de compte endarrere clarament visible amb un senyal acústic automàtic molt potent que sonarà automàticament al final de cada temps de joc, de cada període o període extra.*

- *Durant els últims 60 segons de cada període o període extra, el temps de joc estarà indicat en segons i dècimes de segon."*

Pel que fa al rellotge de possessió (no en tots els casos es obligatòria la seva disponibilitat):

"El dispositiu de 24 segons tindrà un panell de control per operar el dispositiu i monitors amb les especificacions següents:

- *Compte enrera digital indicant el temps en segons.*
- *El monitor en blanc quan cap dels equips tingui control de pilota.*
- *La capacitat d'aturar i tornar a iniciar el compte des del punt en que s'ha aturat."*

I pel que fa als senyals acústics:

"Hauran d'existir com a mínim dos senyals acústics diferents amb sons clarament diferents i molt potents (com per escoltar-se fàcilment en les condicions més adverses o sorolloses):

- *Un pel cronometrador i l'anotador. Pel cronometrador sonarà automàticament per indicar el final de temps de joc d'un període o període extra. Per l'anotador i pel cronometrador es farà sonar manualment quan sigui necessari per cridar l'atenció dels àrbitres sobre la sol·licitud d'un temps mort, d'una substitució, etc. de la finalització del temps mort o de que s'ha produït una situació d'error rectificable.*
- *Un altre per l'operador de 24 segons que sonarà automàticament per indicar el final del compte."*

A nivell professional dos dels auxiliars controlen els rellotges. El cronometrador duu el temps de joc i l'operador de 24 segons gestiona el temps de possessió de pilota. En aquest cas cadascú ha de disposar d'una consola pròpia. A nivell amateur la consola sol integrar cronòmetre i rellotge de 24 segons i ambdós paràmetres són controlats per la mateixa persona.

Fig. 1.1. Auxiliars de taula

En aquest projecte s'ha pretès realitzar el disseny i el muntatge d'un prototip de marcador de bàsquet de petites dimensions degut a l'alt cost de construir-ne un de mides estàndard. L'encariment rau en l'adquisició de *displays* de mides reglamentàries. Dígits de 30 cm tenen un preu que oscil·la entre els 100 i els 120 €. En el seu lloc s'utilitzen *displays* de set segments de menys de 2 cm d'alçada.

Excepte per aquest detall, s'ha respectat el reglament que dicta la FCBQ.

1.1.1 Especificacions

El marcador de bàsquet construït compta amb les següents funcions:

- Cronòmetre de compte enrera pel temps de període (10 minuts inicialment). Minuts i segons poden ser modificats sumant i restant d'un en un. Un cop en marxa es mostren els minuts i els segons fins que queda un minut de temps. A partir de llavors es mostren segons i centèsimes de segon. S'atura, es reprèn i s'inicialitza a 10 minuts a voluntat.

- Cronòmetre de compte enrera pel temps de possessió (24 segons inicialment). Permet suma i resta d'un en un. S'atura, es reprèn i es torna a inicialitzar a 24 a voluntat i independentment del temps de període. S'apaga quan queden menys de 24 segons de temps de partit en cas que es vulgui inicialitzar a 24 per un canvi en la possessió de la pilota durant la jugada.

- Indicador de tempteig local i visitant acumulat (de 0 a 199). Permet suma i resta d'un en un i inicialitzar el compte a zero. Els dígits de les centenes i les desenes romanen apagats mentre no fan falta.

- Indicador del període de joc (d'1 a 8 i "E" per períodes extra). Està controlat per un únic polsador que suma de manera cíclica. Aquest cicle inclou l'estat "apagat".

- Indicador de faltes personals acumulades per cada equip en cada període (de 0 a 5). Està controlat per un únic polsador per equip que suma d'un en un. Un tercer polsador inicialitza les faltes a zero.

- Botzina manual i automàtica. Pot ser activada per un polsador o bé automàticament al final del temps de possessió o del temps de partit.

- Indicador de possessió. Dos leds indiquen a quin equip pertany la possessió de la pilota en un moment determinat segons quin dels dos estigui encès.

- Indicador de direcció de la possessió. Informa de quin equip té la possessió a l'inici dels períodes o després de situació de "salt" indicada per l'àrbitre.

- Indicador de temps morts demanats. Una fila de 3 leds per cada equip indicarà quants temps morts ha fet.

Com es pot observar, les especificacions del marcador compleixen els requeriments del reglament i afegeix informació no disponible en marcadors comercials, com és el cas de la fletxa d'indicació de possessió a l'inici dels períodes o el de mostrar fins a les centèsimes de segon del temps de partit un minut abans del final de cada període. La norma de la fletxa és relativament moderna i encara no es veu reflectida als marcadors. Actualment s'utilitza una cartolina reglamentària amb una fletxa dibuixada que es col·loca a la taula dels auxiliars. Indica el sentit d'atac de l'equip a qui pertoca la possessió.

La pantalla LCD amb la qual compta la consola tampoc es un element comú als marcadors comercials, sobretot entre els menys moderns. Mostra un resum a temps real del contingut del tauler de visualització. Aquesta eina està pensada per facilitar la feina dels auxiliars. Amb ella no cal d'aixecar la vista de la consola per comprovar la reacció en el tauler dels botons que premen.

1.1.2 Diagrama de blocs

El marcador dissenyat en aquest projecte està format per una consola de control amb visualització de dades mitjançant pantalla LCD i pel propi tauler visualitzador.

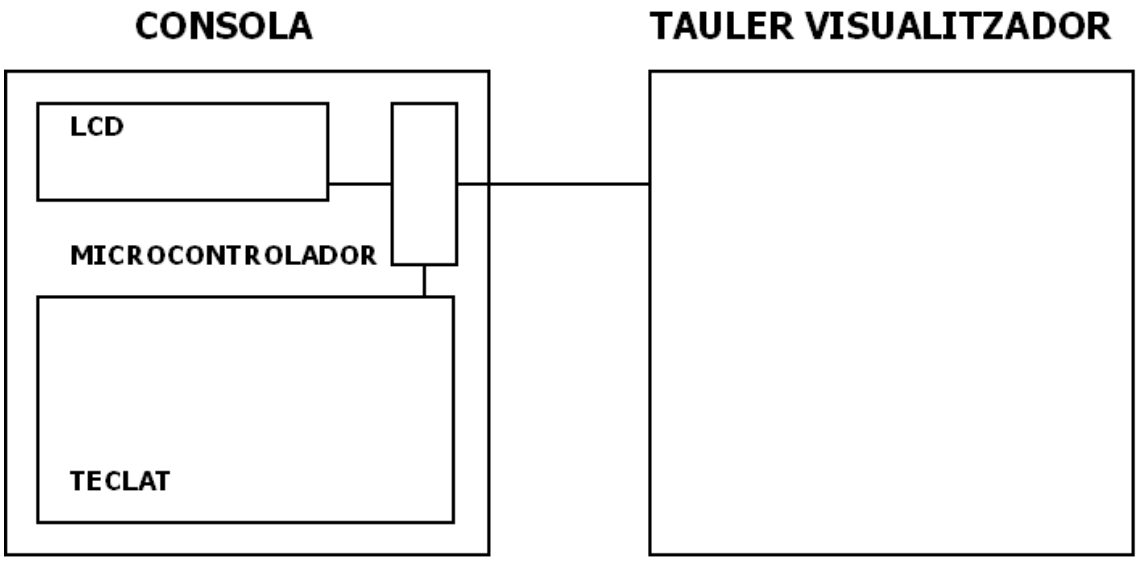


Fig.1.2. Diagrama de blocs del marcador de bàsquet

Típicament la comunicació entre la consola i el tauler ha de permetre que aquests dos estiguin separats per desenes de metres, ja sigui per cable o per ràdio. Això permet la col·locació del tauler en un bon lloc per ser vist des de qualsevol punt de les instal·lacions.

Una altra opció és que consola i tauler estiguin units un darrere de l'altre en el que seria un marcador de taula. En el mercat són molt més barats pel seu menor tamany però el camp de visió es molt més reduït, ja que el públic situat darrere de la taula d'auxiliars i el personal a les banquetes dels equips, a les bandes, no el poden veure. En aquests casos la consola sempre disposa d'una pantalla des d'on l'auxiliar pot veure la informació que mostra el tauler a l'altra banda.

Fig.1.3. Exemple de marcadors de taula

En el prototip d'aquest projecte no es provaran grans distàncies entre consola i tauler.

CAPÍTOL 2. Disseny/muntatge/verificació dels mòduls

2.1 Consola de control

La consola es l'element que comunica la persona amb el tauler de visualització. Ha de presentar un aspecte senzill, intuïtiu i agradable per a la ràpida comprensió del seu funcionament. Cal pensar que l'auxiliar de taula que l'ha de manipular es troba a cada terreny de joc amb diferents marcadors de diferents fabricants i, per tant, amb consoles de tot tipus. L'objectiu del disseny ha de ser facilitar-li la seva tasca.

Fig. 2.1. Exemple de consoles en el mercat

2.1.1 Teclat

El teclat és la part de la consola que permet la comunicació entre l'usuari i el tauler informatiu. La quantitat de botons que inclogui i la manera com estiguin disposats i senyalitzats afectarà directament a la dificultat de comprendre a simple cop d'ull el funcionament i possibilitats del marcador. Pocs botons i palanques commutadores compliquen el seu aprenentatge i un botó per a cada funció li resta dinamisme al seu ús.

Tenint això en compte s'ha pensat en utilitzar un total de 23 polsadors per fer front a les especificacions proposades per la federació, la funció dels quals es detalla a continuació:

Taula 2.1. Funcions dels polsadors de la consola

Polsador	Nom	Descripció
1	RST_TEMPTEIG	restableix a zero el tempteig
2	+1_L	suma un punt a l'equip local
3	-1_L	resta un punt a l'equip local
4	+1_FALTA_L	suma una falta a l'equip local
5	PERIODE	canvia el període
6	POS_L	encén el LED de possessió local
7	POS_V	encén el LED de possessió visitant
8	RST_FALTES	restableix a zero les faltes dels equips
9	CANVI_POS	canvia de sentit la fletxa de possessió
10	+1_V	suma un punt a l'equip visitant
11	-1_V	resta un punt a l'equip visitant
12	+1_FALTA_V	suma una falta a l'equip visitant
13	+1_MIN	suma un minut al temps de període
14	-1_MIN	resta un minut al temps de període
15	RST_CRONO	restableix el cronòmetre a 10 minuts
16	CRONO	inicia/deté el cronòmetre
17	+1_SEG	suma un segon al cronòmetre
18	-1_SEG	resta un segon al cronòmetre
19	ZUMBADOR	activa la botzina
20	+1_POS	suma un segon de possessió
21	-1_POS	resta un segon de possessió
22	RST_POS	restableix la possessió a 24
23	POSSESSIO	inicia/deté el compte de possessió

Una bona distribució de les tecles és essencial per aconseguir un disseny que ajudi a l'auxiliar a comprendre en un breu cop d'ull les possibilitats i funcions del marcador. Per aconseguir-ho, s'han repartit els 23 polsadors en 5 zones diferenciades:

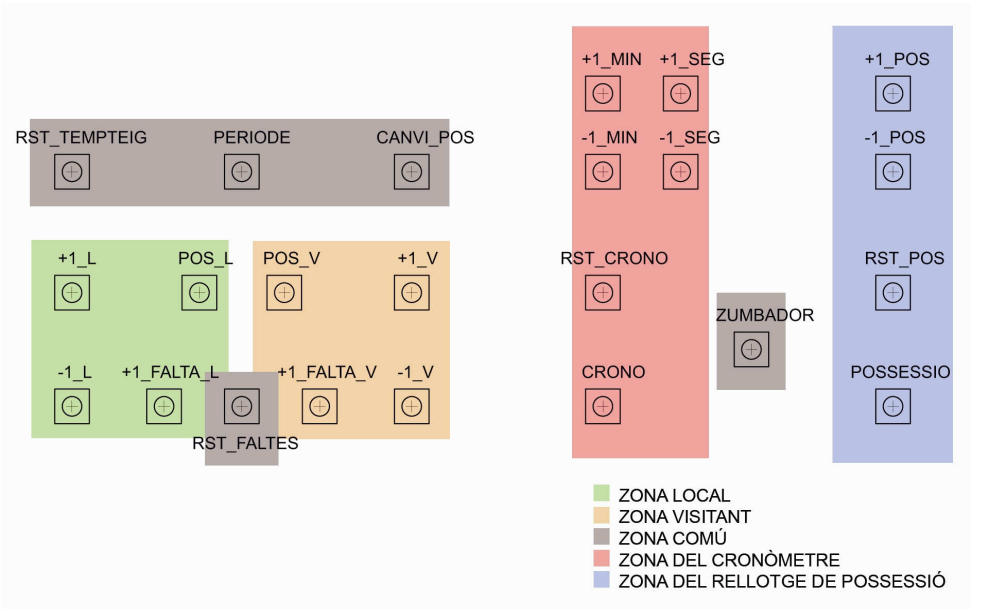


Fig. 2.2. Distribució del teclat per zones

Hi ha exemples de consoles poc intuïtives que degut a que són de les més habituals als pavellons d'esports catalans (segons la FCBQ) són, fins i tot, estudiades als cursos d'auxiliar de taula que la federació imparteix. Es tracta de consoles de marcadors antics que alguns clubs mantenen degut a l'alt cost d'adquirir-ne un de nou. Actualment els preus dels marcadors de nivell amateur van des dels 1.400 € d'un de portàtil fins als 3.300 € (sense comptar transport i instal·lació).

Fig. 2.3. Consola MS2000

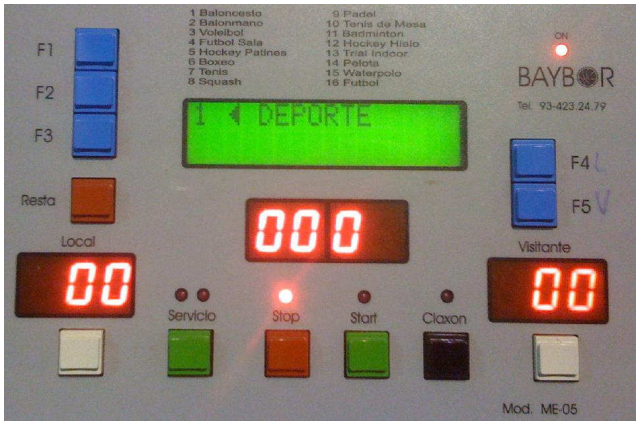


Fig. 2.4. Consola Baybor ME-05

Sembla obvi que la disposició de les palanques i els botons no és gens aclaridora. Fins i tot en el cas de la Baybor ME-05 (figura 2.5) es poden veure anotacions a bolígraf per saber quina es la tecla de l'equip local i quina la del visitant.

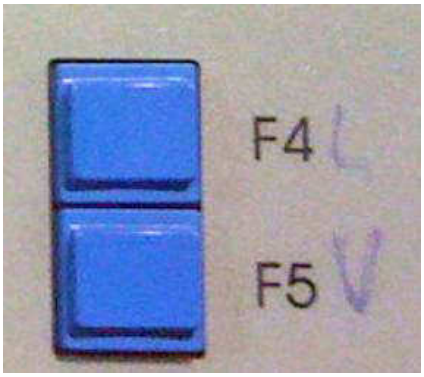


Fig. 2.5. Detall de la figura 2.4

És un efecte que s'intenta evitar al màxim en aquest projecte.

2.1.1.1 P6007

Els polsadors escollits són els P6007, adquiribles a la botiga de components electrònics de Barcelona ONDA RADIO. No apareixen a les llibreries de l'eina EAGLE però es pot descarregar una que té polsadors de la família DTS-6, de les mateixes dimensions que els escollits, des de la pàgina del fabricant del programa (Cadsoft).

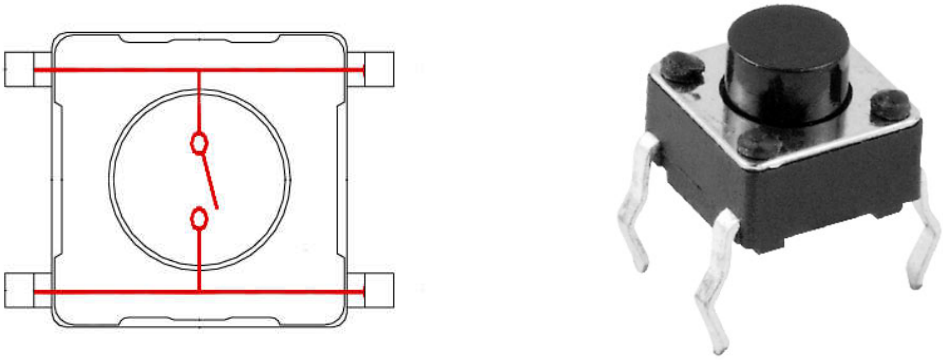


Fig. 2.6. Esquema i imatge del polsador P6007

Són de 4 pins interconnectats dos a dos, de 7 mm d'alçada i de 6x6 mm de costat. Tenen una vida mínima de 100.000 cicles, però s'allarga com més suaument es premen.

Aquest tipus de polsador és els que també té la placa d'assaig per microcontroladors PSoC que s'analitza més endavant en aquest treball.

2.1.2 Expansor de ports

Un cop es pren la decisió d'utilitzar 23 polsadors i es té en compte que tant aquests com la pantalla LCD i els leds de set segments consumeixen pins d'entrada/sortida del microcontrolador, es preveu que sigui quin sigui el microcontrolador que escollim, segurament no proporcionarà la quantitat suficient de pins.

Per aconseguir ports suficients s'aprofita la capacitat dels expansors de ports i s'escull el PCF8574, que permetrà l'estudi i la utilització del bus I2C que comunicarà els expansors amb el microcontrolador, del qual només en consumirà dos pins.

Concretament s'ha escollit el PCF8574P, de 16 pins en format DIP (*Dual In-line Package*, encapsulat de doble línia, també conegut com DIL), que facilitarà la seva soldadura a la placa de circuit imprès pel seu major tamany i separació entre pins respecte a altres tipus d'encapsulat. Altres criteris tinguts en compte per escollir-lo són la seva aparició a les llibreries del programari d'edició

d'esquemes electrònics EAGLE, el qual es farà servir per dissenyar la placa de circuit imprès, i la seva disponibilitat a la botiga de components ONDA RADIO de Barcelona.

2.1.2.1 Bus I2C

Introducció

L'I2C (*Inter-Integrated Circuit*) és un bus de comunicacions en sèrie dissenyat per Philips Semiconductors a principis dels anys 80. En mode estàndard la seva velocitat de transmissió és de 100 Kbps, encara que permet taxes de fins a 3.4 Mbps en casos especials. S'empra molt a la indústria, principalment per comunicar microcontroladors amb els seus perifèrics en sistemes integrats (*Embedded Systems*), que és el cas que es tracta en aquest projecte.

Característiques

La seva principal característica és que empra dues línies per transmetre la informació. Una transmet les dades (anomenada SDA, *System Data*) i l'altra el senyal de rellotge (anomenada SCL, *System Clock*). Ambdues necessiten resistències de *pull-up* (o de polarització) perquè són drenador obert. Cal una tercera línia, la de referència, massa, però no sol ser necessària perquè en la majoria de casos els circuits que es volen comunicar pertanyen a la mateixa placa i ja la comparteixen.

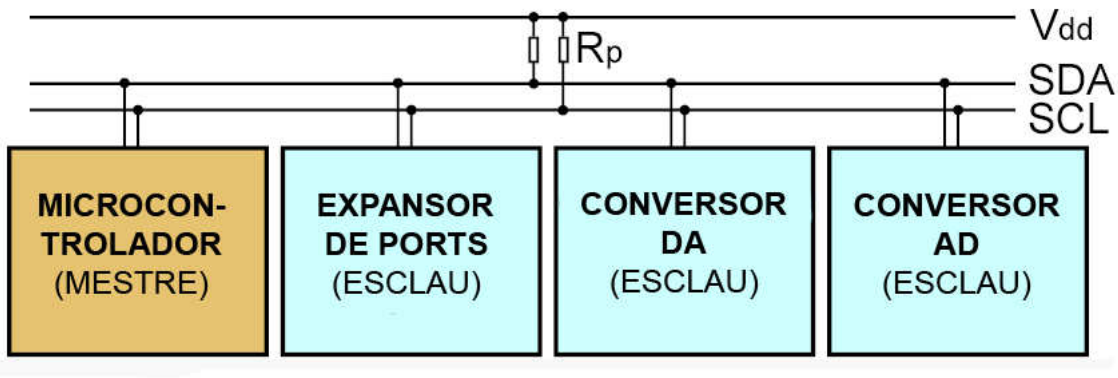


Fig. 2.7. Bus I2C i sistema mestre-esclau

Cada dispositiu connectat al bus I2C disposa d'una adreça única per accedir a ell o per saber de quin d'ells provenen les dades. A més, aquests poden ser mestre o esclau (*master* o *slave*). El dispositiu mestre és l'encarregat d'iniciar la transferència de dades i de generar el senyal de rellotge. No cal que sigui sempre el mateix dispositiu, aquesta característica es pot intercanviar entre dispositius que tinguin aquesta capacitat, en el que s'anomena mode multimestre.

Protocol

El bus està lliure quan les línies SDA i SCL estan en estat lògic alt. Aleshores qualsevol dispositiu pot ocupar el bus com a mestre. Aquest inicia la comunicació enviant un patró (*start condition*, posant a nivell baix la línia de dades) que posa en alerta els dispositius esclau i els deixa a l'espera d'una transacció.

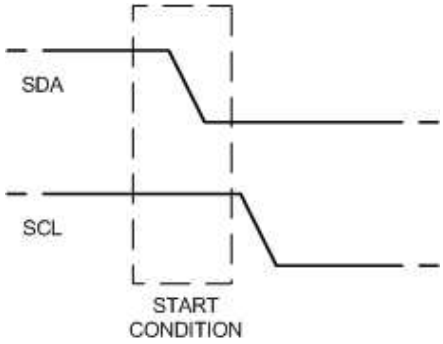


Fig. 2.8. Inici de la comunicació

El mestre es dirigeix al dispositiu amb què es vol comunicar enviant un *byte* que conté 7 *bits* amb l'adreça d'aquest últim i un altre (el de menys pes) que indica escriptura o lectura (enviament o recepció de dades, 0 i 1 respectivament). Aleshores l'esclau compara la direcció rebuda amb la pròpia i en cas de coincidència es considera a si mateix com a esclau-transmisor o esclau-receptor, segons el bit R/W (*Read/Write*, lectura o escriptura). Seguidament l'esclau respon enviant un bit de reconeixement (*acknowledge* o ACK) que indica al mestre que reconeix la sol·licitud i que està en condicions de comunicar-se.

Llavors comença l'intercanvi d'informació entre els dispositius. El mestre envia la direcció del registre intern de l'esclau (si en disposa) que desitja llegir o escriure i aquest retorna un altre ACK. Ara el mestre pot enviar o rebre *bytes* de dades que l'esclau ha de respondre amb un ACK cada vegada.

Encara que el mestre és l'encarregat de generar el senyal de rellotge de la línia SCL, un esclau de menor velocitat pot forçar aquesta línia a nivell baix i fer entrar el mestre en un estat d'espera fins que és capaç de reprendre la comunicació.

Quan la comunicació finalitza, el mestre transmet una condició d'aturada (*stop condition*) per deixar lliure el bus, després de la qual és obligatori que el bus romangui en estat *idle* (aturat, desocupat) durant uns microsegons.

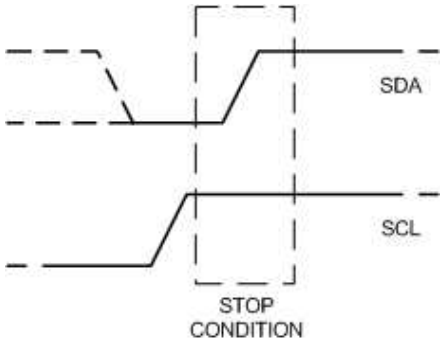


Fig. 2.9. Final de la comunicació

2.1.2.1.1 PCF8574

Introducció

El PCF8574 de Philips és un expansor de ports de 8 bits que es comunica amb altres dispositius mitjançant l'estàndard de comunicació I2C abans explicat. Permet obtenir un guany de 8 ports al sistema al que s'integra. Amb tres dispositius d'aquest tipus es farà front a la necessitat de connectar 23 polsadors al microcontrolador.

Característiques

Una característica atípica que posseeix en comparació amb altres dispositius que utilitzen el bus I2C és una línia d'interrupció (INT), que connectada al microcontrolador pot informar de l'arribada de dades als seus ports sense utilitzar la línia de dades ni la de rellotge. La posa a nivell baix cada cop que es modifica l'estat de qualsevol dels ports.

La correspondència de pins és la següent:

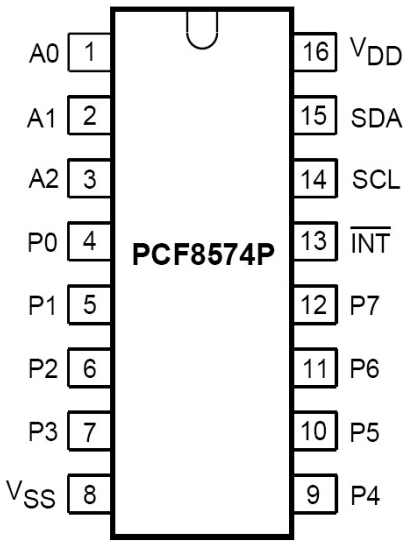


Fig. 2.10. pins del PCF8574P

Taula 2.2. Correspondència de pins

Símbol	Pin	Descripció
A0	1	entrada d'adreça 0
A1	2	entrada d'adreça 1
A2	3	entrada d'adreça 2
P0	4	pin d'entrada/sortida 0
P1	5	pin d'entrada/sortida 1
P2	6	pin d'entrada/sortida 2
P3	7	pin d'entrada/sortida 3
V_{SS}	8	terra
P4	9	pin d'entrada/sortida 4
P5	10	pin d'entrada/sortida 5
P6	11	pin d'entrada/sortida 6
P7	12	pin d'entrada/sortida 7
INT	13	sortida d'interrupció
SCL	14	línia de rellotge
SDA	15	línia de dades
V_{DD}	16	alimentació

Funcionament

El seu adreçament funciona, com en el cas d'altres dispositius que permeten comunicació I2C, amb una paraula de 7 bits. En aquest cas concret els 4 primers bits més rellevants són fixes (0100 en el cas del PCF8574P) i els tres darrers, A2, A1 i A0 són modificables i es connecten a nivell alt o baix. Això

permet utilitzar 8 direccions diferents, des de la 0100000 a la 0100111 (32 i 39 en decimal).

Un cop generada la *start condition* el mestre ha d'enviar al bus el byte amb l'adreça de 7 bits de l'esclau desitjat seguit del bit d'escriptura/lectura. L'expansor al·ludit enviarà un ACK com a reconeixement. Aquest expansor no disposa de registres interns, directament modifica l'estat dels seus ports (en cas d'escriptura) o n'informa (en cas de lectura). Si la intenció era escriure, el mestre enviarà un byte indicant quins ports vol actius (1) i quins inactius (0). En cas de lectura, l'esclau enviarà al mestre el byte amb l'estat actual dels 8 ports. Aquests bytes tenen la forma P7 P6 P5 P4 P3 P2 P1 P0, on cada bit informa de l'estat del port corresponent.

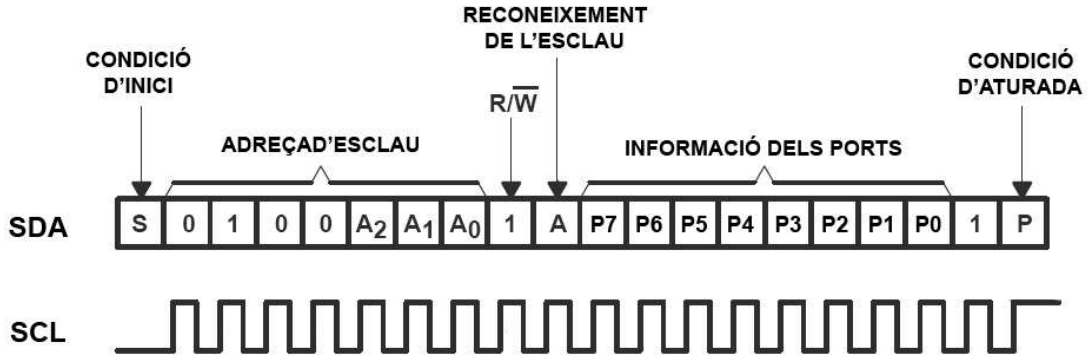


Fig. 2.11. Exemple de comunicació amb el PCF8574

En el cas d'aquest projecte interessa llegir l'estat de cada port, o sigui, l'estat dels polsadors de la consola. Quan s'alimenta el PCF8574, els pins passen a nivell alt. Per tant, si es fes una lectura de l'expansor, s'obtindria la paraula 1111 1111. En cas que es connectés un polsador connectat a terra a cada port de l'expansor, se'n pitgés el connectat al P3 i es fes una lectura en aquell moment, s'obtindria la paraula 1111 0111 i es deduiria el polsador concret que ha sigut pitjat.

2.2 Tauler visualitzador

El tauler és l'element més visible del marcador. La seva finalitat és mostrar les dades del partit i que aquesta informació arribi a tots els punts del recinte esportiu.

Com en el cas de la consola, en la qual s'han emprat expansors de ports degut a la previsió de falta de ports al microcontrolador, s'utilitzen controladors de leds per reduir el nombre de línies necessàries i simplificar el disseny.

2.2.1 LED display driver

Per controlar els leds de set segments es poden utilitzar circuits integrats dissenyats especialment per a aquest fi, que simplifiquen molt el disseny del circuit. En aquest projecte es fa servir el *MAX7219 led display* driver, capaç de controlar fins a 8 dígits o bé, fins a 64 leds convencionals (en matriu de 8x8) i de comunicar-se amb altres dispositius mitjançant 3 o 4 línies, segons el cas, mitjançant el bus de comunicació SPI. Així doncs, amb el mateix tipus de dispositiu es controlen els 15 dígits i els 20 leds convencionals necessaris pel tauler.

2.2.1.1 Bus SPI

Introducció

L'SPI (*Serial Peripheral Interface*, batejat per Motorola) és un estàndard de comunicacions emprat per la transferència d'informació entre circuits integrats en equips electrònics, igual que l'I2C. Els dispositius també es comuniquen en mode mestre/esclau.

Característiques

Consta de 4 línies, una de rellotge (SCLK), una de sortida de dades, una d'entrada de dades (les dues últimes anomenades MISO, *Master In Slave Out*, o MOSI, *Master Out Slave In*, segons es tracti d'un mestre o un esclau) i una última de selecció de dispositiu (*slave select*, SS, o *chip select*, CS). Això comporta dos desavantatges clars enfront l'I2C: consumeix més terminals en cada xip i necessita una línia de *slave select* per cada dispositiu que es connecti al bus. Altres desavantatges són la dificultat d'aplicar el mode multimestre i que no s'utilitza reconeixement i que, per tant, el mestre podria enviar informació sense un esclau connectat i no adonar-se'n.

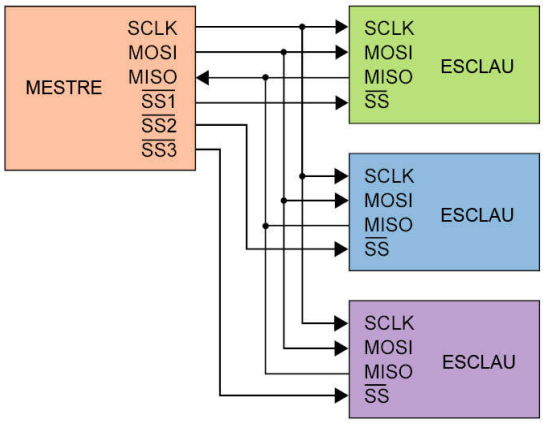


Fig. 2.12. Exemple de bus SPI

Malgrat això té punts a favor considerables com la comunicació *full duplex* (enviament i recepció simultània de dades), una major velocitat de transmissió i un menor consum d'energia.

Protocol

La comunicació no comença sense la configuració del senyal de rellotge per part del mestre, que en aquest cas permet freqüències d'entre 1 i 70 MHz. També permet establir la polaritat d'aquest i la seva fase respecte dels senyals de transmissió/recepció de dades.

Per escollir l'esclau al que es vol adreçar posa a nivell baix la línia SS que els connecta. A vegades aquest requereix un temps d'espera que el mestre ha de respectar abans de començar a transmetre el senyal de rellotge.

Durant cada cicle de rellotge es produeix una comunicació *full duplex* encara que no sigui necessària. Els bits passen per *buffers* (files d'espera) d'entrada i sortida de tamany preestablert abans de ser transmesos o rebuts i abocats a la línia o als registres interns quan són plens.

La comunicació s'atura quan no hi ha més dades a transmetre. El mestre atura el rellotge i deixa lliure la línia SS de l'esclau amb què es comunicava.

2.2.1.1.1 MAX7219

Introducció

El MAX7219 és un controlador de leds de set segments de càtode comú compatible amb els estàndards de comunicació SPI, QSPI i Microwire. En aquest projecte es farà servir per connectar leds, tant individuals com de set segments, al microcontrolador mitjançant bus SPI.

En concret s'ha escollit el MAX7219CNG, d'encapsulat de tipus DIP, de 24 pins, que consta a les llibreries de l'eina EAGLE i també es pot adquirir a ONDA RADIO.

Característiques

Treballa amb una tensió d'entre 4 i 5,5 volts i suporta freqüències de rellotge de fins a 10 MHz.

Els seus registres interns permeten, entre altres possibilitats interessants, modes d'encesa de tots els leds connectats, regular la intensitat lluminosa dels leds, actualitzar dígits de manera individual i codificar-los en BCD.

Fa possible un cas de connexió SPI poc comú en què només cal una línia d'SS per tots els esclaus, l'anomenada *daisy chain* (cadena de margarides). Això permet estalviar línies de connexió entre els controladors de leds i el

microcontrolador. En el cas que es tracta es connecten 3 controladors en cascada per gestionar els 15 dígits i els 20 leds individuals del tauler visualitzador.

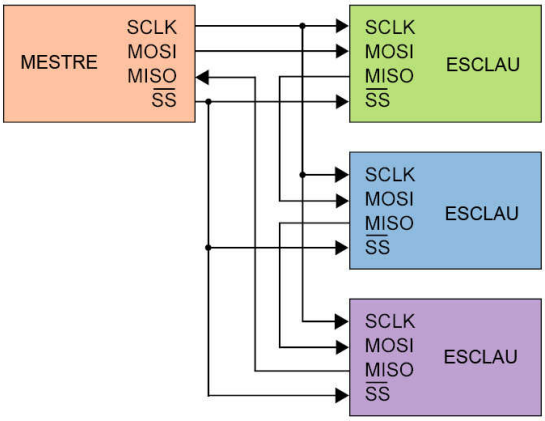


Fig. 2.13. Exemple de connexió en cadena

La correspondència de pins és la següent:

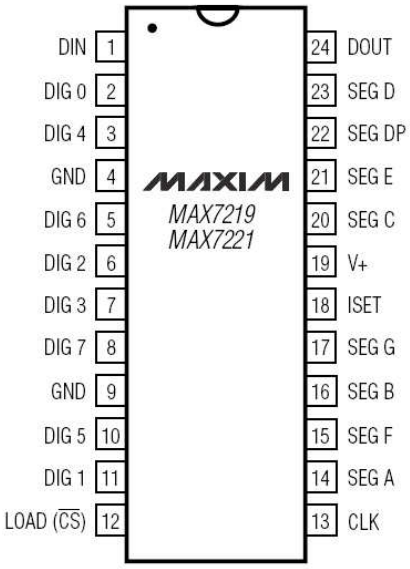


Fig. 2.14. Pins del MAX7219

Taula 2.3. Correspondència de pins

Símbol	Pin	Descripció
DIN	1	Línia d'entrada de dades. Aboca els bits de dades al registre intern de desplaçament de 16 bits a cada flanc de pujada del rellotge.
DIG 0-DIG 7	2, 3, 5-8, 10, 11	Línies que alimenten cada dígit.
GND	4, 9	Terra. Ambdós pins han d'estar connectats entre ells.
LOAD	12	Aboca els últims 16 bits de dades durant el flanc de pujada.
CLK	13	Entrada del senyal de rellotge. Durant el flanc de pujada un bit de dades entra al registre de desplaçament. Durant el de baixada un bit de dades surt per DOUT. Taxa màxima de 10 MHz.
SEG A-SEG G, DP	14-17, 20-23	Línies que proporcionen el corrent a cada segment i al punt decimal dels leds.
ISET	18	Es connecta a la línia de voltatge a través d'un resistor (R_{SET}) per establir el màxim corrent dels segments.
V^+	19	Entrada d'alimentació positiva. Connectar a +5 V.
DOUT	24	Línia de sortida de dades. Les dades a DIN arriben a DOUT 16.5 cicles de rellotge després. S'utilitza per concatenar diversos MAX7219.

Funcionament

La comunicació funciona mitjançant paraules de 16 bits que mantenen el format següent:

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
X	X	X	X	ADREÇA				MSB	DADES						LSB

Fig. 2.15. Format dels paquets de dades

on els bits del D15 al D12 són bits sense significat, del D11 al D8 conformen l'adreça del registre intern al que es vol accedir i del D7 al D0 són el tipus d'acció que es vol executar en el registre escollit.

Els bits de dades van entrant al registre de desplaçament del dispositiu al ritme del senyal de rellotge. Un cop omplertes les 16 posicions és necessari un canvi d'estat del senyal LOAD, de nivell baix a nivell alt, just després del 16è flanc de pujada del rellotge. Durant el flanc de pujada del senyal LOAD els 16 bits entren simultàniament als registres interns. Els bits sense significat no entren enlloc, del D8 al D11 entren al descodificador d'adreça de registre i del D7 al D0 passen a un registre de dades (figura 2.16).

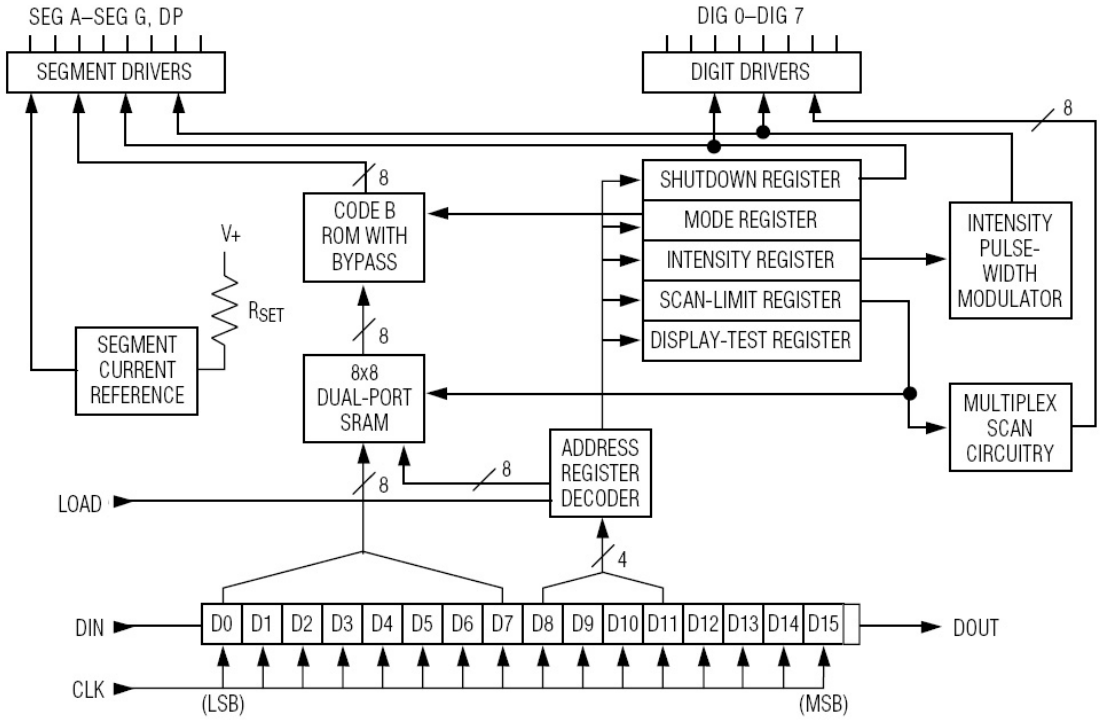


Fig. 2.16. Diagrama de blocs del MAX7219

Tipus de registres

El MAX7219 compta amb 7 tipus de registres que es detallen a continuació.

- Registre d'apagada. Aquest registre disposa de dos estats, el mode d'apagada i el mode normal. Durant el mode d'apagada, tots els *displays* connectats romanen apagats sense alterar la configuració dels registres. És el mode en el qual es troba el MAX7219 inicialment quan s'alimenta. Se n'ha de sortir activant el mode normal per mostrar la informació que es vol reflectir als leds.

- Registre de prova de dispositiu. També té dos estats, el de prova i el normal. El de prova també ve predeterminat quan s'alimenta el dispositiu i es mostra quan es desactiva el mode d'apagada. Encén tots els segments respectant també la configuració de la resta de registres. Per sortir-ne cal activar el mode normal.

Encara que els modes d'apagada i de prova són els predeterminats no impedeixen programar de manera prèvia a la seva desactivació els registres que venen a continuació.

- Registre d'escombrat. Amb aquest registre es determina quants dígits són utilitzats d'1 a 8 (si no es configura només s'activa el primer). En el cas d'utilitzar diversos MAX7219 es aconsellable utilitzar el mateix nombre de leds a cada un o almenys configurar aquest registre com si fos així per tal que uns no semblin més brillants que els altres.

- Registre d'intensitat. Té la capacitat de gestionar la intensitat de lluentor dels segments dels dígits i dels leds. Aquest paràmetre pot ésser controlat també pel valor de la resistència R_{SET} connectada entre l'alimentació del dispositiu i el pin I_{SET}. Segons el full d'especificacions el seu valor mínim hauria de ser 9,53 kΩ, amb el qual s'obté un valor del corrent als segments de 40 mA ja que segons consta a les especificacions és 100 cops el corrent que passa per I_{SET}. A partir d'aquest valor es pot fer disminuir la brillantor amb un potenciòmetre en sèrie amb R_{SET}.

- Registre de dígit. Selecciona amb quin dígit, del 0 al 7, es vol operar. Els 8 bits de dades segueixen un format o un altre segons s'utilitzi descodificació BCD o no.

- Registre de descodificació. Permet indicar quins dels dígits utilitzats fa servir o no codi BCD. En cas d'activar-lo, el bit D7 indica l'estat de la coma decimal del dígit (DP), els bits del D6 al D4 no tenen significat i del D3 al D0 indiquen la xifra en codi BCD que es vol que mostri el dígit. També és possible representar els caràcters H, E, L i P, a més del guió intermedi i de deixar el dígit en blanc. En cas de no utilitzar el descodificador, del bit D7 fins al D0 es controla els estats de la coma decimal i dels segments de l'A al G respectivament.

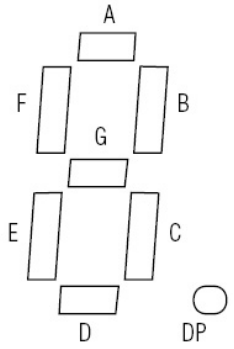


Fig. 2.17. Segments del led

- Registre de "no operació". S'utilitza quan es connecten diversos MAX7219 en cascada. Permet no alterar els dispositius amb els que no es desitja operar mentre s'opera amb un altre.

Connexió en cascada

El MAX7219 permet connexió en cascada en cas de necessitar controlar més de 8 leds de set segments. Això s'aconsegueix compartint els senyals de rellotge i LOAD i connectant la sortida DOUT de cada dispositiu amb l'entrada DIN del següent.

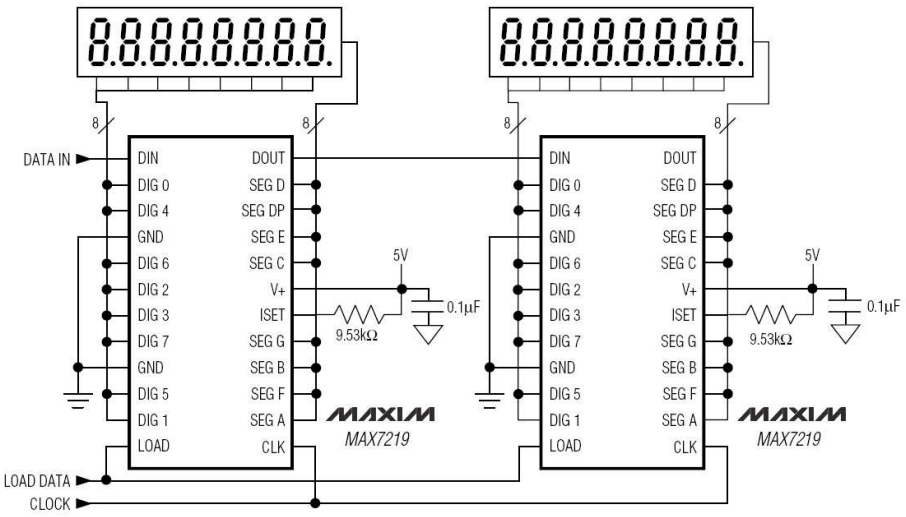


Fig. 2.18. Dos dispositius en cascada

En aquest tipus de connexió es important la possibilitat de “no operació” que proporciona el registre del mateix nom. Per exemple, si tres dispositius MAX7219 són connectats en cascada i es vol comunicació amb el segon cal enviar els 16 bits de dades que es vol enviar al segon dispositiu entre dos codis de “no operació”. Aleshores, provocant un flanc de pujada en el senyal LOAD, els codis de “no operació” entren al primer i al tercer dispositius (quedant inalterats) i el segon rep les dades que provoquen els canvis oportuns.

En el cas d’aquest projecte es necessiten tres MAX7219 connectats en cascada. El primer controlarà 8 leds de set segments, el segon en controlarà 7 i el darrer gestionarà 20 leds ordinaris.

2.2.2 Leds de set segments

Per a la representació visual de les dades al tauler s’escull la utilització de dispositius LED, tant en forma de visualitzador de set segments com en forma convencional. Els de set segments seran del tipus de càtode comú a causa de les característiques del MAX7219.

La disposició dels dispositius és la de la figura de la dreta.

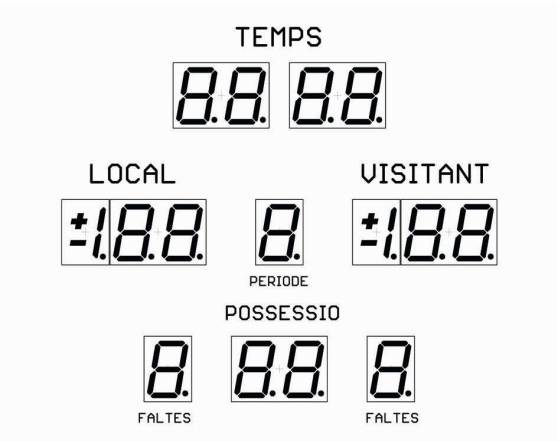


Fig. 2.19. Disposició dels leds al tauler

Els models escollits es detallen a continuació.

2.2.2.1 MAN6940

Aquest és un model de doble dígit que s'aprofita per mostrar els minuts i els segons del temps de període, els segons del temps de possessió i les unitats i desenes del tempteig dels dos equips.

Té les següents característiques:

- Alçada de dígit de 14,2 mm.
- Color vermell d'alta eficiència.
- Càtode comú.
- Especial per ambients lluminosos.
- Angle de visió de 150º.
- Alta brillantor.
- Alt contrast.

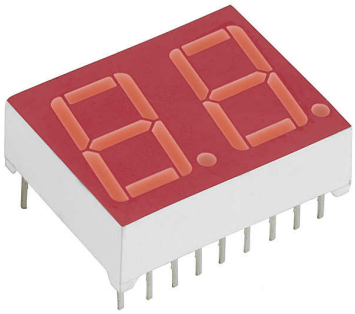


Fig. 2.20. MAN6940

2.2.2.2 HDSPH103

Model de dígit individual, s'utilitza per indicar l'acumulació de faltes dels dos equips i el període en joc.

Té les següents característiques:

- Alçada de dígit de 14,2 mm.
- Color vermell.
- Càtode comú.
- Angle de visió de 100º.

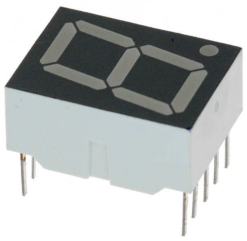


Fig. 2.21. HDSPH103

2.2.2.3 HP5082-7756

Aquest model permet representar el dígit 1 amb signe i coma decimal. Serveix per mostrar les centenes del tempteig acumulat dels dos equips.

Té les següents característiques:

- Alçada de dígit de 10,9 mm.
- Color vermell d'alta eficiència.
- Mode universal (càtode i ànode comú en un mateix dispositiu)
- Especial per ambients lluminosos.
- Alta brillantor.
- Alt contrast
- Visible fins a 5 metres.

Fig.2.22. HP5082-7756

2.3 Microcontrolador

El microcontrolador és el cervell dels equips electrònics que l'incorporen com els microones, els telèfons, els televisors, el sistema d'arrancada dels cotxes i, com en aquest cas, els marcadors electrònics. Rep les dades de la consola, les processa i les plasma als elements visuals.

Per una correcta elecció del microncotrolador s'ha de tornar a pensar en els requeriments inicials i en el tipus de perifèrics que s'han anat escollit durant el disseny. Un resum de les necessitats que ha de satisfer pot ser el següent:

- Alimentació similar a la dels perifèrics.
- Dos rellotges independents de velocitats diferents.
- Capacitat per comunicació I2C i SPI.
- Pins d'entrada/sortida suficients per a la comunicació amb els perifèrics.
- Programable en un llenguatge conegut i mitjançant un programari senzill.
- Encapsulat apropiat per la possible soldadura.
- Adquisició factible, senzilla i barata.

Cal tenir en compte també que els rellotges s'han de poder configurar a la velocitat dels segons i de les centèsimes i que dins d'aquestes resolucions de temps el microcontrolador ha de poder executar una sèrie d'instruccions per actualitzar els *displays* numèrics a temps.

S'ha decidit triar-ne un de la marca Cypress MicroSystems, el PSoC (Programable System on-Chip), perquè compleix amb tots els requeriments anteriors i a més és innovador, versàtil, dinàmic, barat i ideal per iniciar-se en el món dels microcontroladors.

No es tracta encara d'un dels tipus més populars de microcontroladors, com podrien ser els PIC, els AVMEL o els AVR (de recursos més estàtics), però la seva fama va en augment.

Per trobar el PSoC més adient per a un projecte es pot fer servir un document disponible a la pàgina web del fabricant que inclou una guia de selecció de dispositius segons les necessitats que hagi de satisfer.

2.3.1 PSoC

El PSoC és un microcontrolador molt versàtil i totalment dinàmic que permet disposar de tota una sèrie de components electrònics com filtres, amplificadors, comparadors, conversors analògic-digital i digital-analògic, comptadors i temportitzadors de 8, 16 i 32 bits, interfícies de comunicació (com UART, I2C o SPI) etc. sense necessitat d'augmentar el tamany del circuit final, ja que els incorpora en el seu interior. Aquest dinamisme és la seva característica diferencial vers altres microcontroladors més populars però estàtics.

Consta de 2 tipus de blocs, els analògics i els digitals, on poder col·locar l'ampli ventall de components abans esmentats i que es poden connectar als pins d'entrada/sortida del PSoC mitjançant línies internes. La quantitat de blocs varia segons la família de PSoC.

També posa a l'abast de l'usuari una sèrie de senyals de rellotge internes, derivades de la senyal principal de 24 MHz, que es poden utilitzar de manera simultània.

Una altra característica important és el controlador d'interrupcions de què disposen tant els blocs com la tensió d'alimentació, el senyal de rellotge i els pins d'entrada/sortida. Proporciona un mecanisme que, en complir-se una certa condició, interromp l'execució del programa principal i passa a executar un nou codi específic. Quan l'ha executat, retorna al punt on estava.

Per programar aquests components Cypress posa a l'abast de l'usuari programari específic i gratuït. Pel cas de la família PSoC1, a la qual pertany el microcontrolador que s'utilitza en aquest projecte, existeixen el PSoC Designer 5 i el PSoC Programmer 3.10. Són les versions més noves actualment i ambdues han estat utilitzades en l'elaboració d'aquest projecte.

El PSoC és programable en llenguatge assemblador però també en C (principal avantatge respecte els famosos microcontroladors PIC), més senzill i flexible que el primer. Per programar en C s'ha de comprar una llicència però hi ha la opció d'utilitzar una versió *lite* gratuïta que redueix en un 40% el ventall de possibilitats de la versió original però que pel cas d'aquest projecte no ha suposat un greu problema.

2.3.2 CY8C29466

Seguint les indicacions de la guia de selecció de PSoC abans esmentada, s'arriba a la conclusió que el CY8C29466-24PXI satisfarà les necessitats del marcador. Té les següents característiques:

- S'alimenta amb una tensió d'entre 4,75 i 5,25 volts, semblant a la dels expansors de ports i la dels controladors de leds.
- Compta amb 32 KB de memòria Flash i 2 KB de SRAM que se suposen més que suficients per a contenir el programa que permetrà controlar el marcador.
- Treballa a una freqüència de 24 MHz.
- Consta de 16 blocs digitals i 12 analògics.
- Es comunica i alimenta a través de 28 pins en format DIP.

- 24 dels 28 pins totals estan repartits en 3 ports de 8 bits cada un i es poden configurar com a pins d'entrada o de sortida.

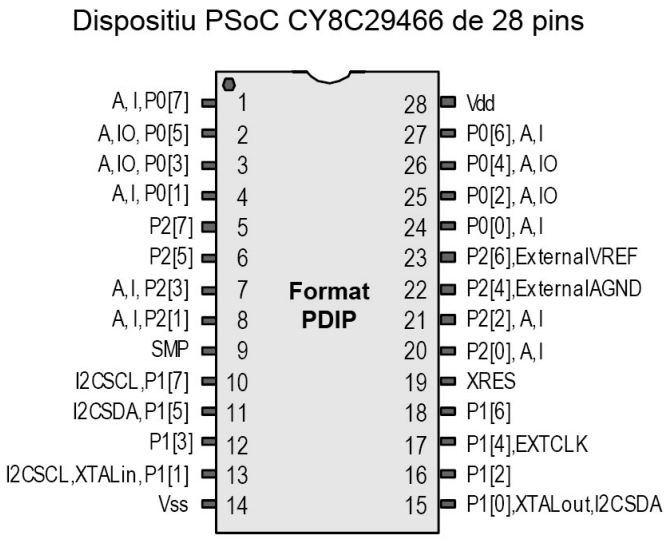


Fig. 2.23. Correspondència de pins del CY8C29466

2.3.3 PSoC Designer i PSoC Programmer

PSoC Designer és l'encarregat d'emplaçar i programar els components als blocs. La versió 5, com a novetat respecte a versions anteriors, permet dos tipus de procés de disseny, a nivell de xip (*chip level*) i a nivell de sistema (*system level*).

El primer permet configurar minuciosament el dispositiu i és el que s'ha fet servir en aquest projecte. Està pensat per usuaris que tenen nocions de programació i de microcontroladors. Aquest procés segueix els passos següents:

- Creació d'un nou projecte.
- Tria del PSoC que es vol utilitzar.
- Tria i configuració dels mòduls (perifèrics interns).
- Connexió dels mòduls entre ells i amb els pins d'entrada/sortida del PSoC.
- Redacció del codi en C o assemblador.
- Programació del PSoC i comprovació del seu funcionament.

El segon està pensat per aquells que volen prescindir de la redacció de codi de programació. Funciona a través d'una interfície de programació visual senzilla que permet configurar el dispositiu. Segueix aquests passos:

- Creació d'un nou projecte
- Selecció de les entrades i les sortides.

- Definició del comportament d'aquestes.
- Simulació i verificació del disseny.
- Elaboració automàtica del programa.
- Programació del PSoC i comprovació del seu funcionament.

Un cop dins de l'entorn del chip level apareix una interfície amb una sèrie de marcs:

- Esquema de l'arquitectura interna del PSoC: mostra els blocs digitals i analògics que conté el microcontrolador, les línies internes amb les quals es poden interconnectar o connectar amb els ports i els propis ports.

- Recursos globals: lloc on configurar característiques del PSoC tals com voltatge d'alimentació, velocitat de treball de la CPU o freqüències de diferents fonts derivades de la freqüència de treball.

- Propietats del mòdul: aquí es mostren els paràmetres i els recursos dels mòduls seleccionats quan se'ls ha emplaçat als blocs.

- Configuració de pins: permet definir el comportament i les característiques dels pins.

- Explorador del projecte: permet navegar entre les carpetes del projecte creat i obrir els arxius que contenen en una nova pestanya.

- Mòduls: aquí es troba tot el llistat d'elements que es poden col·locar als blocs. Cada mòdul disposa del seu full d'especificacions detallat on s'explica el funcionament i les instruccions en llenguatges C i assemblador necessàries per programar-lo. Inclou també exemples de codi i enllaços a la pàgina de Cypress on trobar exemples de projectes funcionals.

Des de la finestra de l'explorador es por accedir al fitxer *main.c*, ubicat a la carpeta *Source Files*, que contindrà el codi de programació. El menú de la pestanya que s'obre quan es selecciona permet accions com detectar errors de codi o crear el fitxer *.hex* que cal enviar al microcontrolador per que funcioni.

Altres fitxers també han de ser modificats a mesura que es van afegint mòduls al projecte, per exemple per fer que una interrupció faci que s'executi una funció concreta del programa.

El menú de la pestanya que conté el main.c permet l'accés al programa PSoC Programmer. Aquest programa comunica l'ordinador amb el PSoC per programar-lo enviant-li l'arxiu *.hex* creat pel PSoC Designer. Per fer-ho cal col·locar el PSoC en una placa d'entrenament, per exemple la CY3210-PSoCEVAL1, i connectar la placa a l'ordinador a través d'un cable USB i el mòdul MiniProg1. Aleshores cal seleccionar el tipus de PSoC i la família a la qual pertany per poder iniciar la transferència de dades.

2.3.4 CY3210-PSoCEVAL1

El CY3210-PSoCEVAL1 és una eina d'assaig per PSoC que inclou:

- placa d'assaig
- cable USB 2.0
- mòdul MiniProg1
- dispositiu PSoC CY8C29466-PXI
- CD d'instal·lació dels programes PSoC Designer i Programmer
- Manual d'iniciació
- petit assortit de cables

Fig. 2.24. Imatge del kit

La placa d'assaig inclou un mòdul LCD per visualitzar els resultats dels assaigs a més de leds, un potenciómetre, un polsador i una placa de forats per la connexió de components electrònics.

Amb ella es poden fer les verificacions necessàries per al disseny i muntatge del marcador i, fins i tot, per tal d'aprofitar la pantalla LCD que incorpora, s'inclou al prototip i forma part de la consola.

2.4 Verificacions

Abans de donar per vàlid cap disseny es procedeix a la comprovació de que les parts del marcador funcionen correctament. És indispensable verificar que s'ha entès el funcionament de tots els dispositius i que es connecten de manera adient abans de passar a la construcció de les plaques, punt on seria molt difícil corregir un error de disseny.

2.4.1 Verificació de la consola

Per comprovar que el material escollit serveix per crear una consola de control pel marcador es crea un disseny simplificat de la mateixa. El que es pretén és verificar que els expansors es comuniquen correctament amb el PSoC a través d'un bus I2C.

Es fa servir la placa d'entrenament amb la pantalla LCD, una protoboard, un parell d'expansors, dos polsadors per cada un i un resistor per cada polsador per limitar el corrent que entrarà als expansors. Cada polsador realitzarà una funció que es veurà reflectida a la pantalla LCD.

Per comprovar que el microcontrolador distingeix correctament els polsadors mitjançant la lectura de les adreces dels expansors, es connectaran dos a dos als mateixos pins dels seus corresponents expansors.

Es recrea la tasca d'acumular el tempteig en un partit de bàsquet. Dos dels polsadors sumaran punts als dos equips i els altres dos en restaran.

2.4.1.1 Esquema del circuit

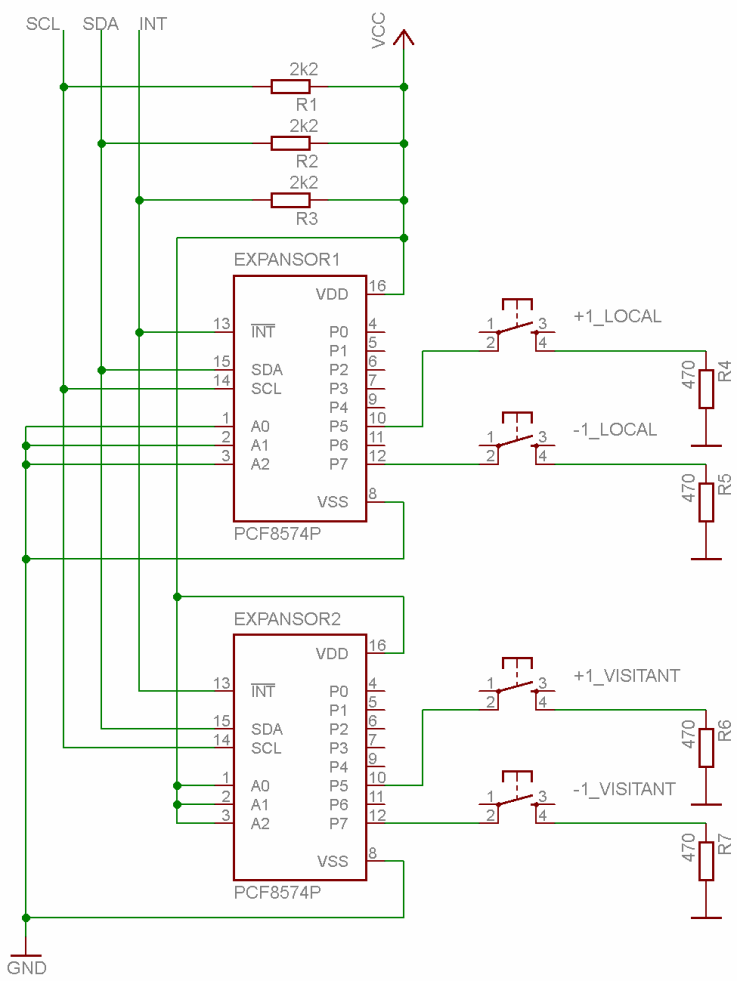


Fig. 2.25. Esquema del circuit de verificació

Les línies SCL i SDA es connecten als pins del PSoC escollits en el menú de configuració del mòdul de comunicació I2C del PSoC Designer 5. La línia INT es connecta a un pin que generi una interrupció cada cop que detecti un canvi de nivell alt a nivell baix.

2.4.1.2 Diagrama de flux

El programa principal de la verificació comença definint les variables generals del programa sencer i donant-li un valor inicial a les que sigui necessari.

Seguidament la pantalla LCD s'activa i mostra la informació necessària, en aquest cas el tempteig acumulat dels dos equips.

L	O	C	A	L				V	I	S	I	T	A	N	T
	0	0											0	0	

En aquest punt s'habiliten les interrupcions generals i les GPIO (s'activen en modificar-se l'estat dels pins d'entrada, o sigui, en aquest cas, en un canvi d'estat del senyal INT dels expansors). Això fa que el programa executi el codi de la interrupció desitjada.

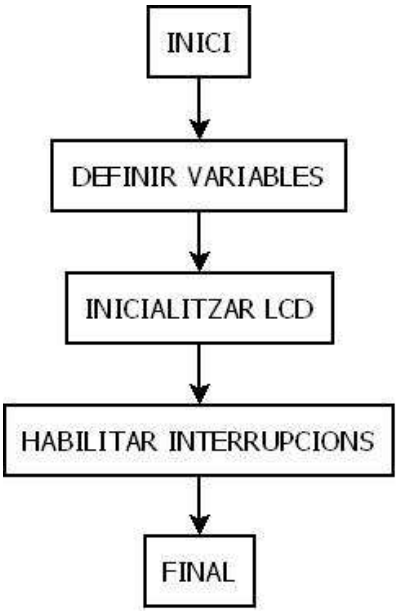


Fig. 2.26. Programa principal

Un cop dins de la interrupció es comprova que s'hi ha entrat degut al canvi d'estat del senyal INT de l'expansor de ports PCF8574 i no per cap altre possibilitat. Aleshores es llegeix la informació de les entrades de cada expansor per comprovar quin dels polsadors ha activat la interrupció. Segons quin d'ells hagi estat premut, una de les dues variables incrementa o decrementa el seu valor en una unitat i s'actualitza el canvi a la pantalla LCD.

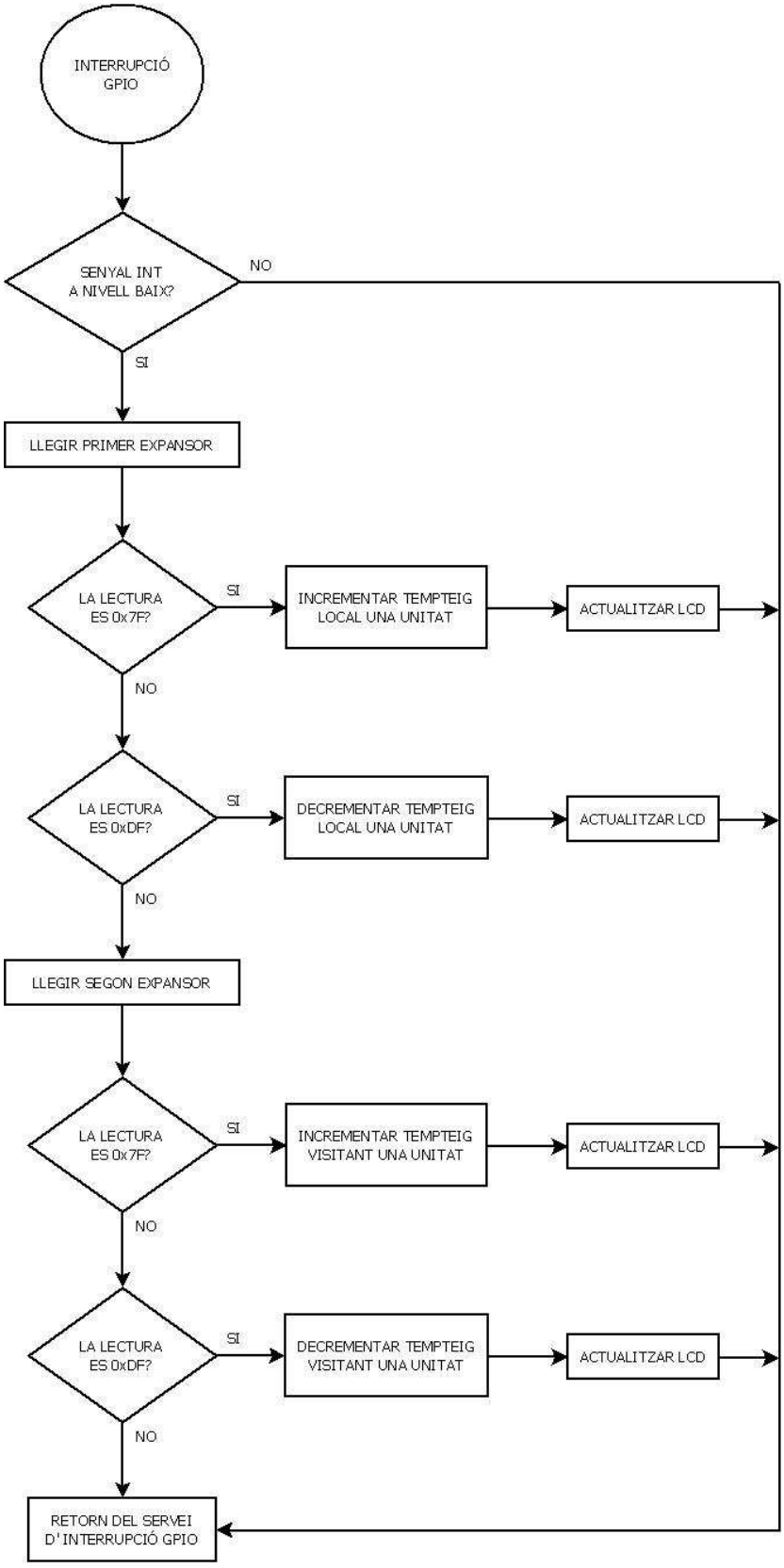


Fig. 2.27. Codi de la interrupció GPIO

2.4.1.3 Resultats

Els resultats són satisfactoris. Cada polsador provoca l'acció que es pretenia. D'aquesta manera s'arriba a la conclusió de que el disseny del teclat és possible si es basa en el d'aquesta prova.

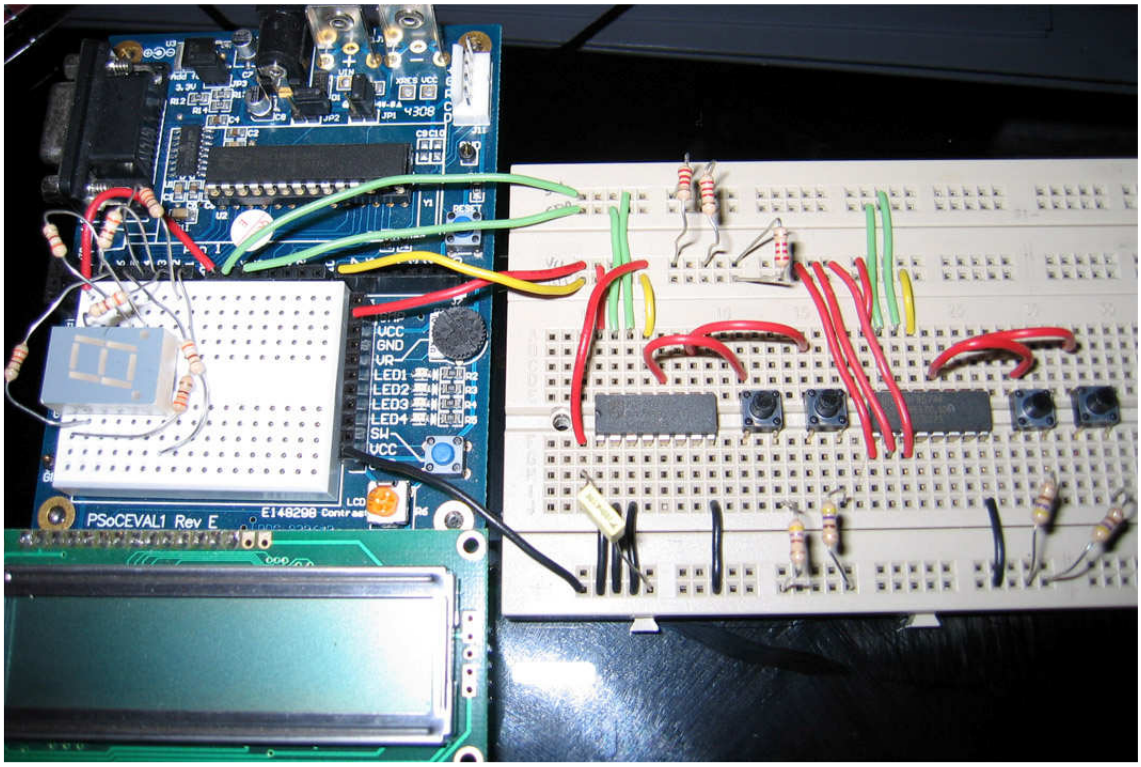


Fig. 2.28. Muntatge de la validació

2.4.2 Verificació del tauler

Per determinar el correcte funcionament dels controladors de leds que formen part del tauler visualitzador es crea un petit exemple de comunicació entre el microcontrolador i els controladors.

Com en el cas de la consola, per dur a terme aquesta prova es fa servir la placa d'entrenament, la protoboard, els drivers de leds, els leds i altres components necessaris com polsadors i resistors.

Es recreen els dos rellotges del marcador amb dos *displays* dobles MAN6940 controlats per un MAX7219 cada un. D'aquesta manera es comprova el muntatge en cascada dels controladors de leds i la independència entre els rellotges que permet el microcontrolador.

Un dels rellotges recrearà el temps de possessió. Actuarà com un compte enrera des de 24 fins a 0 a una velocitat d'1 Hz. Un polsador iniciarà i detindrà

el compte i un altre el restablirà a 24. L'altre rellotge recrearà les centèsimes del temps de període. Comptarà enrera des de 99 fins a 0 a una velocitat de 100 Hz. Dos polsadors més serveixen, com en el cas del temps de possessió, d'inici/aturada i de restabliment a 99.

2.4.2.1 Esquema del circuit

Les línies de rellotge i de transferència de dades es connecten al microcontrolador, als pins que estan connectats internament amb el mòdul de comunicació SPI. La línia que transmet dades del MAX7219 al microcontrolador es deixa sense connectar ja que aquesta comunicació no és necessària.

El senyal LOAD es connecta a un pin que el fa passar de nivell baix a nivell alt després de la transmissió dels bits de dades que es desitja que entrin als registres dels dos controladors.

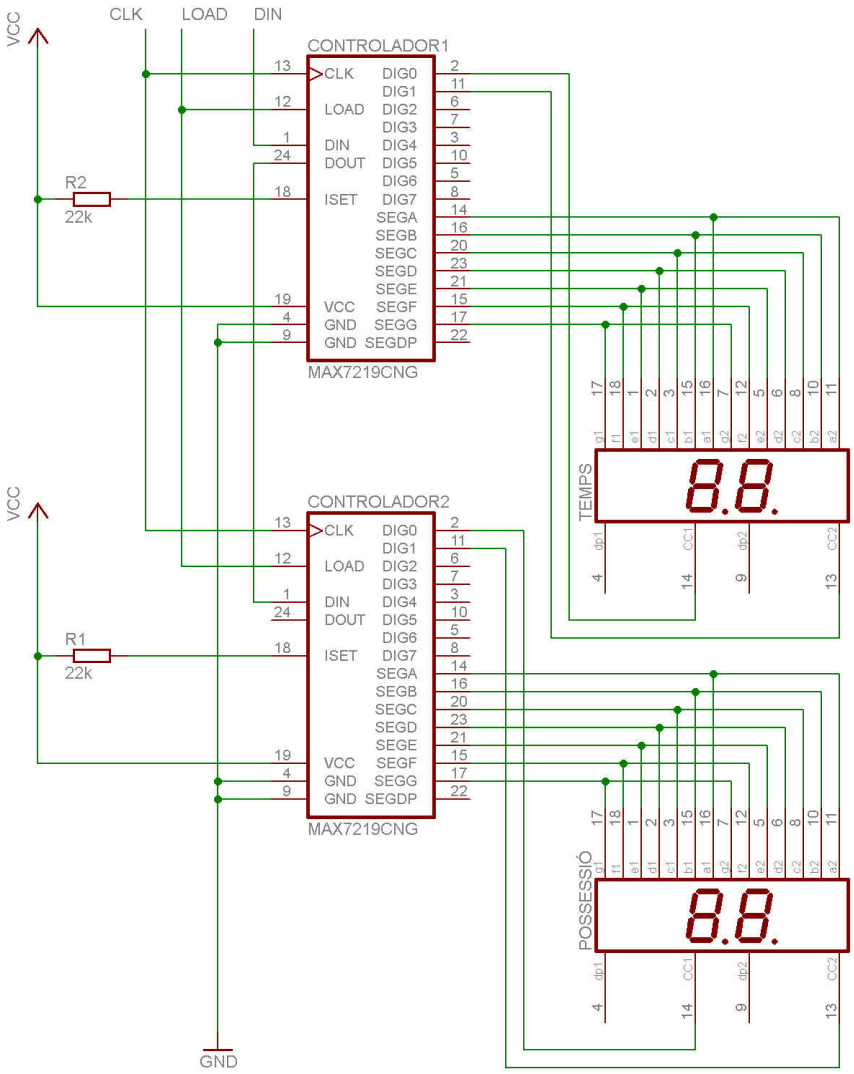


Fig. 2.29. Esquema del circuit de verificació

2.4.2.2 Diagrama de flux

Com en la verificació anterior, el primer pas del programa principal es definir les variables i l'últim és habilitar les interrupcions generals i les GPIO. Ara, a més a més, s'han d'activar les interrupcions dels dos temporitzadors que es necessiten.

Un cop definides les variables s'activen els leds sortint dels modes d'apagada i de prova dels dos controladors i configurant-los degudament per representar un "99" al primer led doble i un "24" al segon.

Després de les habilitacions el microcontrolador està preparat per executar el codi de les interrupcions.

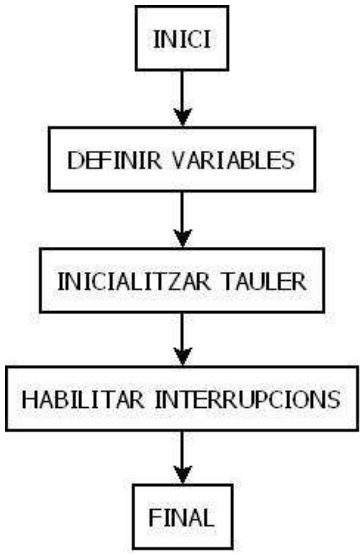


Fig. 2.30. Programa principal

Un cop dintre del codi d'interrupció GPIO es comprova un per un el polsador que ha estat premut i que ha posat un dels senyals d'entrada a nivell alt (és així perquè els polsadors van directament al microcontrolador i s'ha configurat la interrupció d'aquesta manera, en el cas dels expansors el polsador que es premia posava la línia a nivell baix). En trobar el polsador causant de la interrupció s'executa l'acció corresponent i es surt del servei (figura 2.31).

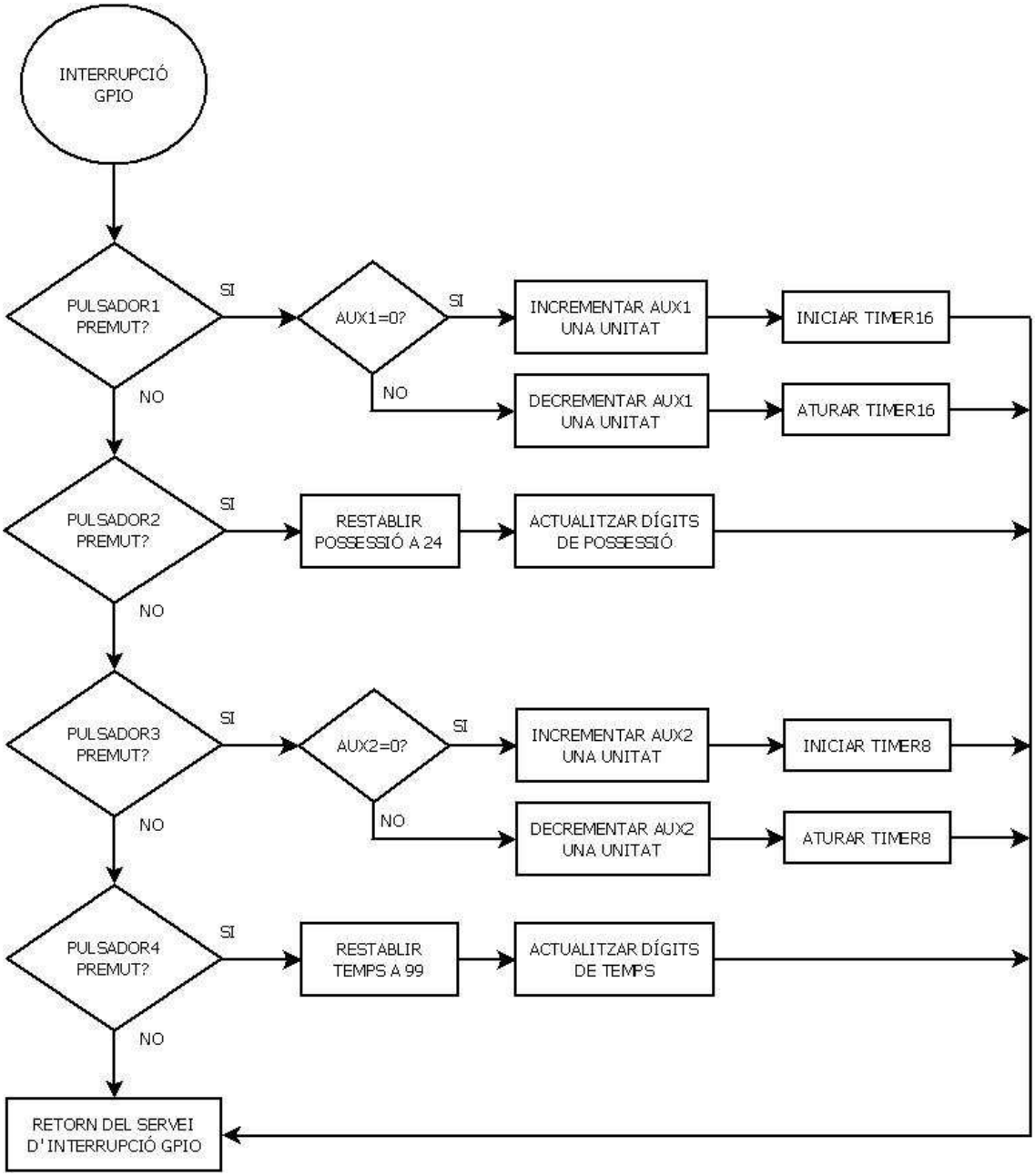


Fig. 2.31. Codi de la interrupció GPIO

Quan s'activa un temporitzador, el programa executa les instruccions que conté la seva funció interrupció a cada pols del rellotge del temporitzador. Es comprova si el temps ha arribat a zero i si no ho ha fet decrementa les variables i actualitza el seu valor als leds de set segments (figures 2.32 i 2.33).

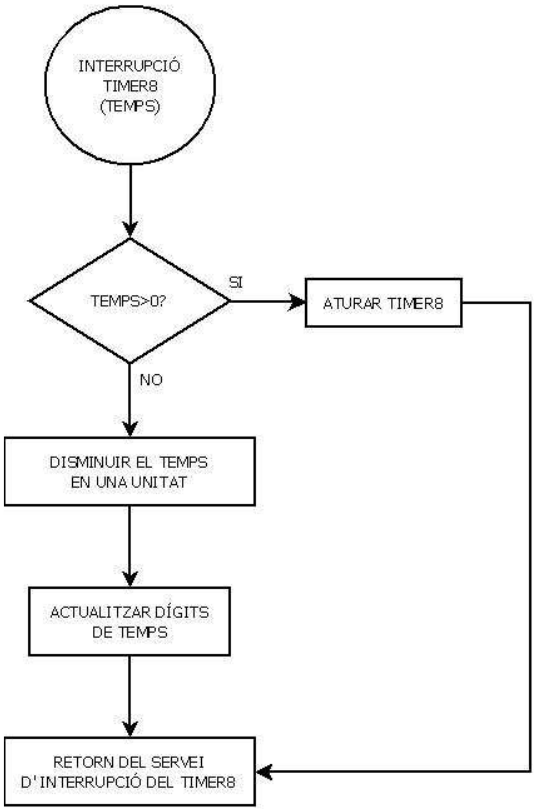


Fig. 2.32. Interrupció del Timer8

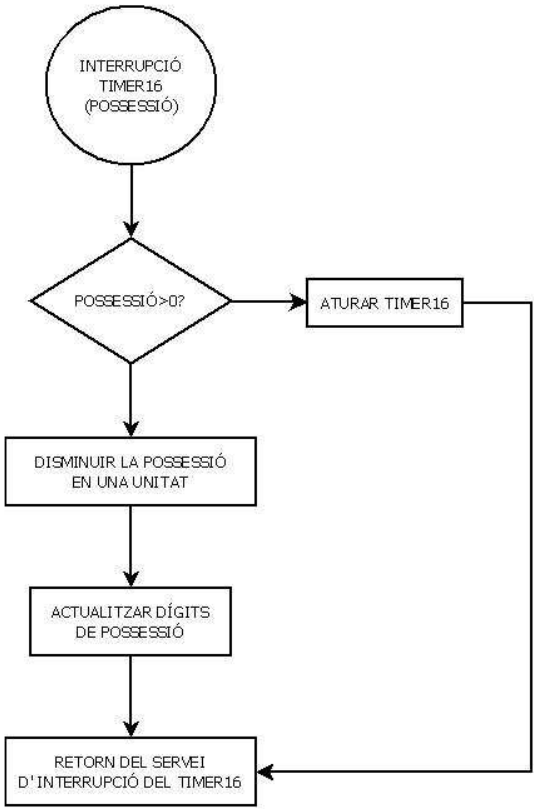


Fig. 2.33. Interrupció del Timer16

Per comprovar que el ritme dels dos comptes concorda es pot fer que no s'aturin mai. Això s'aconsegueix substituint el bloc de "Aturar Timer" dels diagrames de flux anteriors per un que restableixi els comptes. D'aquesta manera es pot comprovar si cada 100 centèsimes del temps de partit el compte de possessió es decrementa un segon i que aquesta relació es manté en el temps. Una darrera comprovació és comparar els ritmes dels rellotges amb el d'un cronòmetre digital professional.

2.4.2.3 Resultats

Els resultats tornen a ser satisfactoris. Després de diverses proves s'observa que els ritmes dels dos rellotges guarden la relació 1/100 durant un interval de temps suficient per considerar que serveixen per controlar el temps d'un partit de bàsquet. Es constata que a més aquests ritmes concorden amb els d'un cronòmetre digital professional.

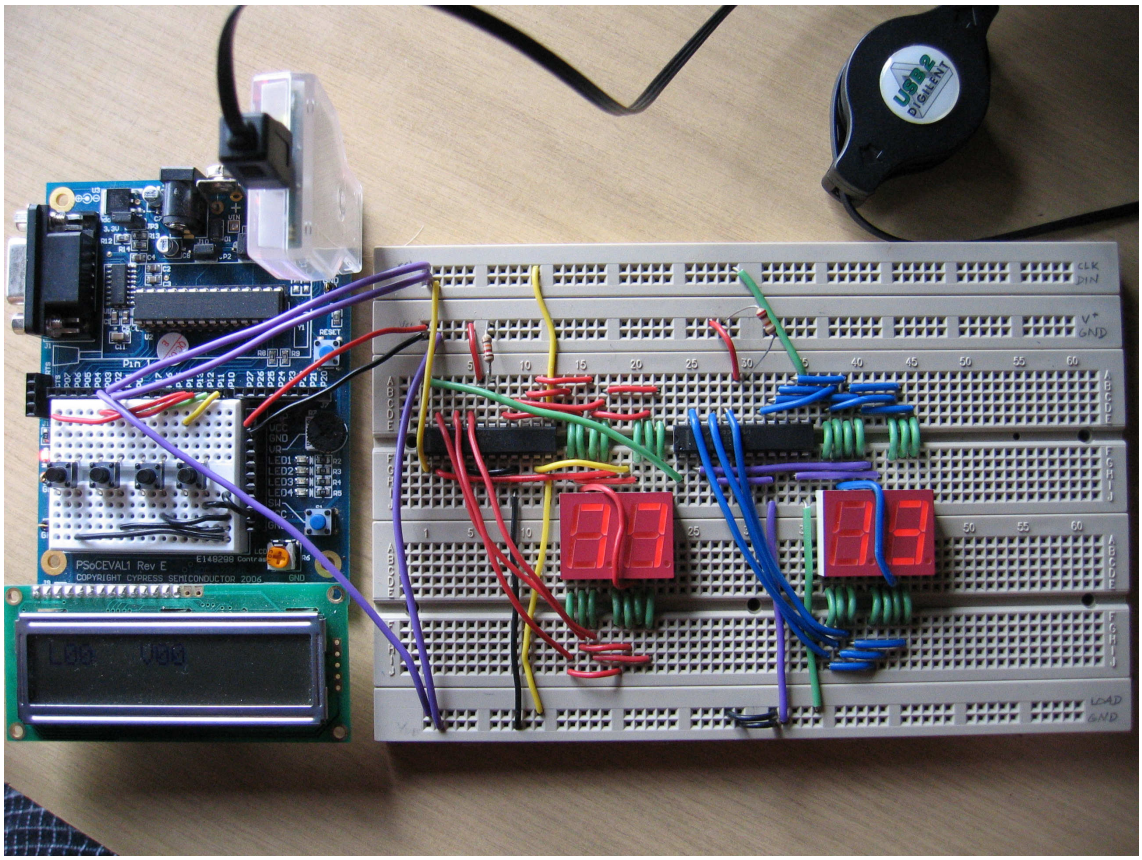


Fig. 2.34. Muntatge i resultat de la verificació

Gràcies a les verificacions s'han pogut aclarir dubtes i corregir idees errònies sobre alguns aspectes del funcionament dels dispositius, i en resum, aconseguir una idea molt clara i acurada de com dissenyar i programar el marcador de bàsquet definitiu.

També han permès agafar pràctica tant en el funcionament i la programació d'un microcontrolador del tipus PSoC com en les possibilitats de la resta d'eines utilitzades.

En aquest punt es pot preveure que si es basa el disseny i la construcció del marcador en els dissenys d'aquest apartat, aquest satisfarà les especificacions inicials de manera correcta.

CAPÍTOL 3. Disseny de l'aplicació completa

Un cop fetes les verificacions que s'han proposat a l'últim apartat del capítol anterior es comença el disseny del marcador de bàsquet complet.

Com ja s'ha comentat amb anterioritat, s'ha fet servir l'eina EAGLE 5.6 tan per confeccionar els esquemes dels circuits com per crear els dissenys de les plaques de circuit imprès.

EAGLE permet, un cop fet l'esquema del circuit, fer un disseny per a la creació d'una placa de circuit imprès mantenint els dispositius i les connexions de l'esquema. Els dispositius i components s'extreuen d'una col·lecció molt completa de llibreries dels fabricants més habituals. En cas de no trobar algun dispositiu es poden trobar més llibreries a Internet o bé, coneixent les seves dimensions, se'n pot crear una amb el mateix programa.

Disposa d'un *Autorouter* que dibuixa de manera automàtica les pistes que connecten els dispositius. Si el circuit és complex el dibuix no sol ser òptim i a partir d'ell s'han de fer modificacions a mà.

Per fer els diagrames de blocs i de fluxos s'ha utilitzat el Dia 0.97, un programa lliure que permet dibuixar esquemes estructurats.

3.1 Esquema del circuit

S'ha dividit l'esquema del circuit en dues parts: l'esquema de la placa de la consola i el de la placa del tauler visualitzador.

3.1.1 Consola

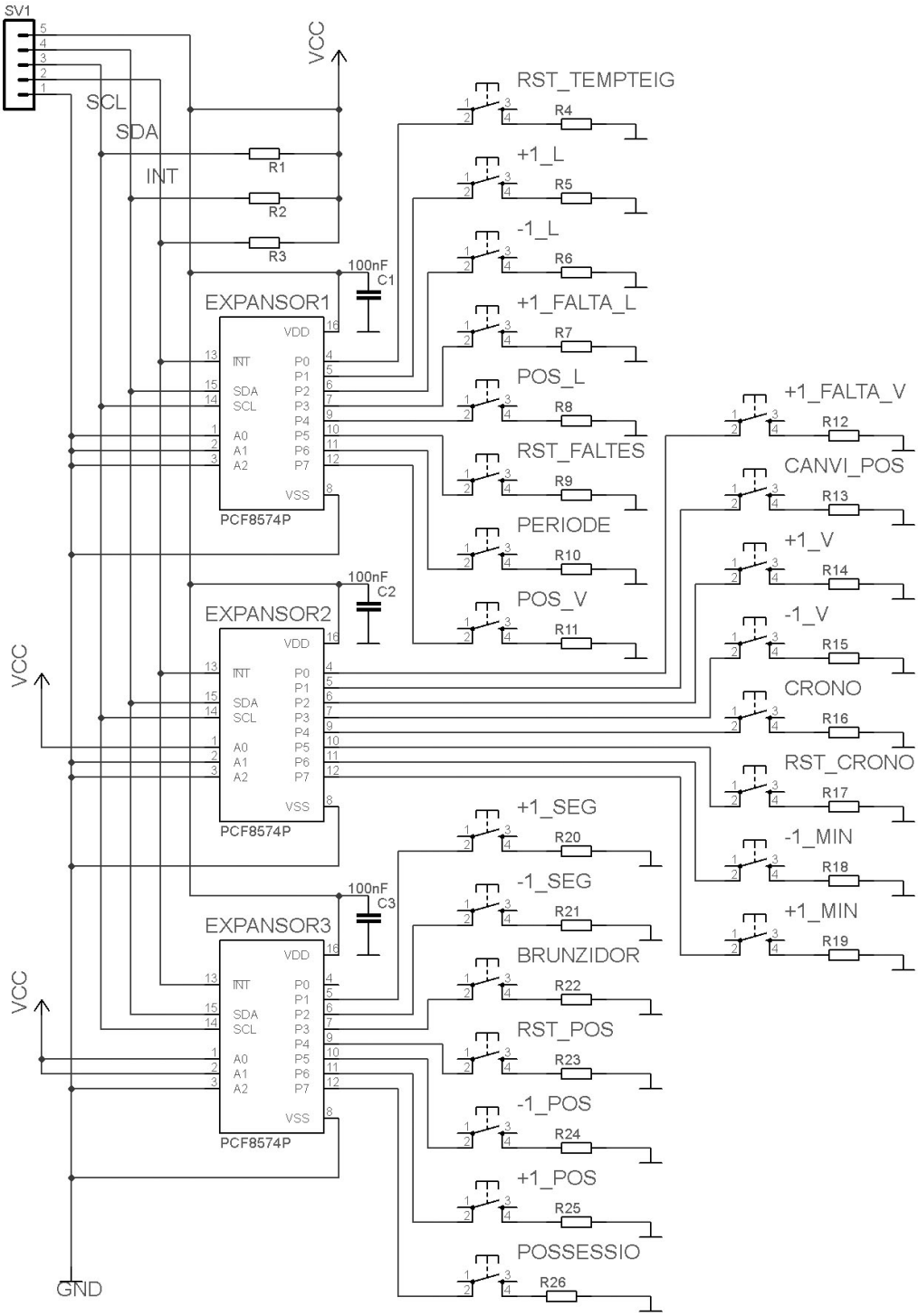


Fig. 3.1. Esquema de la consola

De l'esquema de la figura 3.1 es poden deduir els tres últims bits de l'adreça de cada expansor (A2, A1 i A0). La del primer expansor és 000, la del segon és 001 i la del tercer és 011.

3.1.2 Tauler Visualitzador

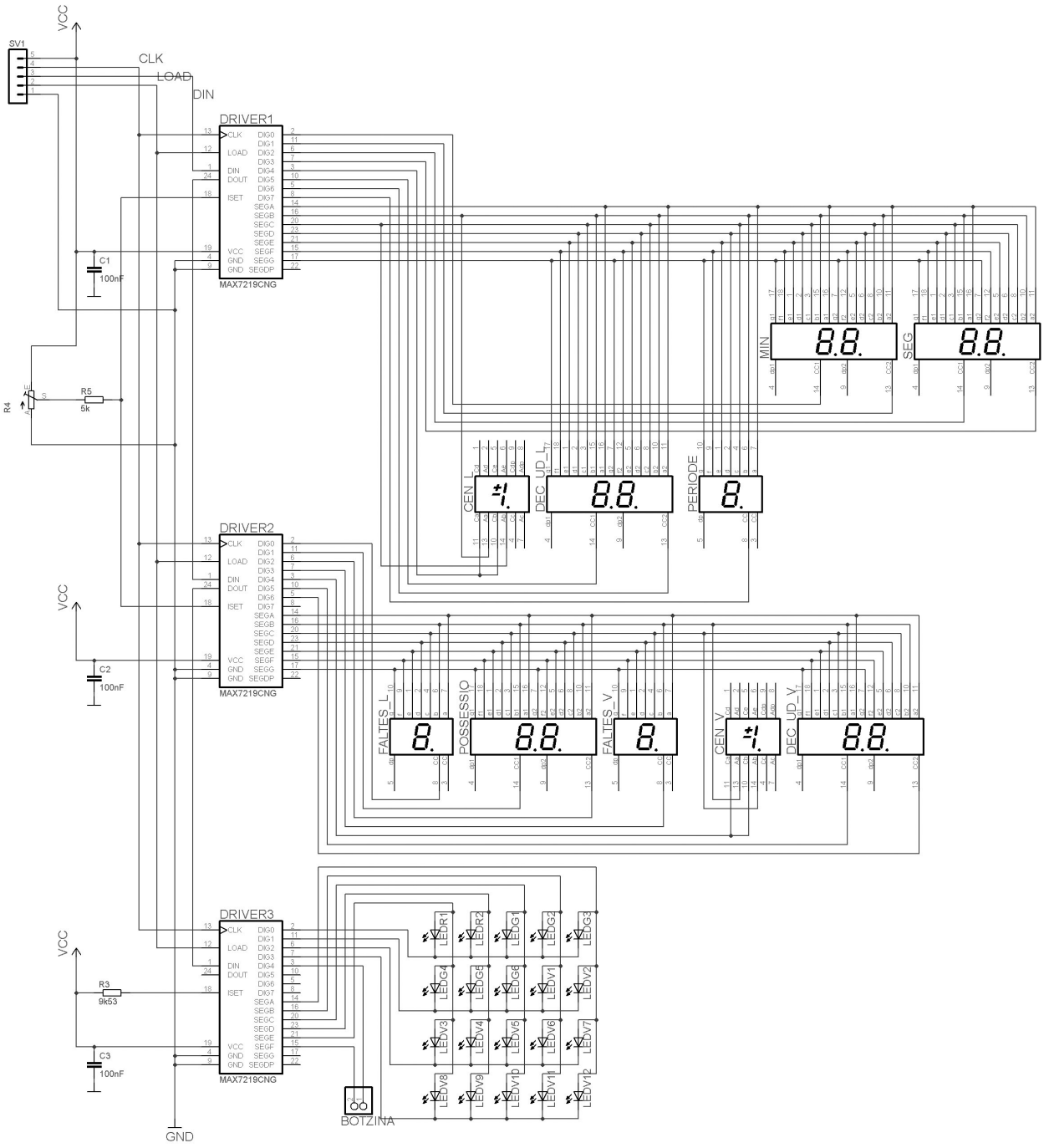


Fig. 3.2. Esquema del tauler visualitzador

A l'esquema final del tauler visualitzador se li ha afegit un potenciòmetre en sèrie amb la R_{SET} que va al driver1 i al driver2 per tal de controlar la intensitat de llum dels dígits.

A més s'han aprofitat els pins lliures del driver3 per controlar una petita botzina.

3.2 Esquema PCB

3.2.1 Consola

La simplicitat de l'esquema de la consola ha permès el disseny de les pistes de la placa pràcticament per una sola cara. Normalment es fa que les pistes passin per la capa inferior (*bottom*, en blau), sobretot si els dispositius utilitzats són de format DIP. Això permet col·locar-los a la capa superior (*top*, en vermell) i soldar-los per sota.

També s'ha fet servir la capa superior per afegir text aclaridor de la funció de cada polsador i de les zones en què s'han agrupat.

Els forats marcats a les cantonades serveixen per recolzar la placa en potes de plàstic o en cargols.

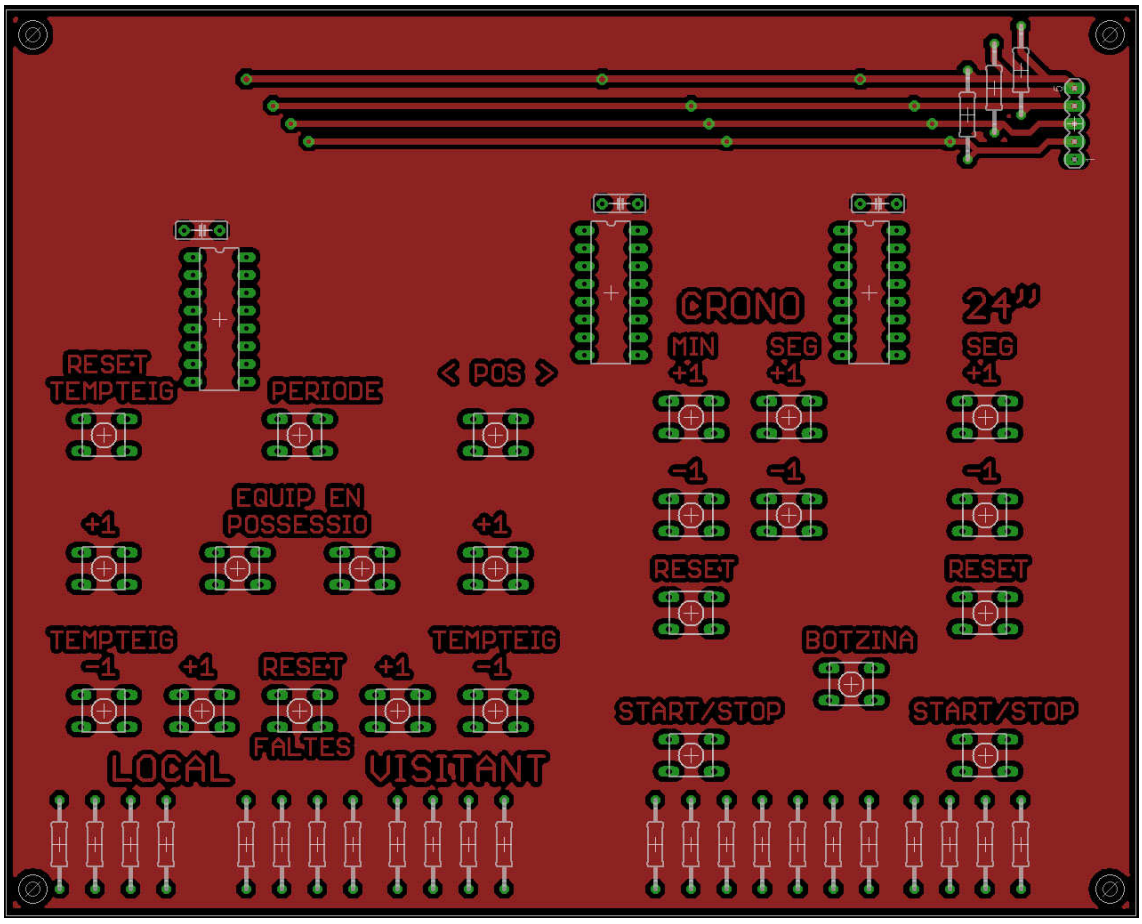


Fig. 3.3. Capa superior de la placa de circuit imprès de la consola

En l'esquema de la placa també s'identifiquen els 23 polsadors i els 3 expansors de ports. La pantalla LCD l'aprofitarem de la placa d'entrenament del CY8C29466, que forma part de la consola i conté també el microcontrolador.

A la part superior dreta es pot veure una fila horitzontal de 5 sòcols. Connecten les línies d'alimentació, SDA, SCL, INT i massa a la consola, concretament a la placa d'entrenament, que com ja s'ha dit és on està ubicat el microcontrolador.

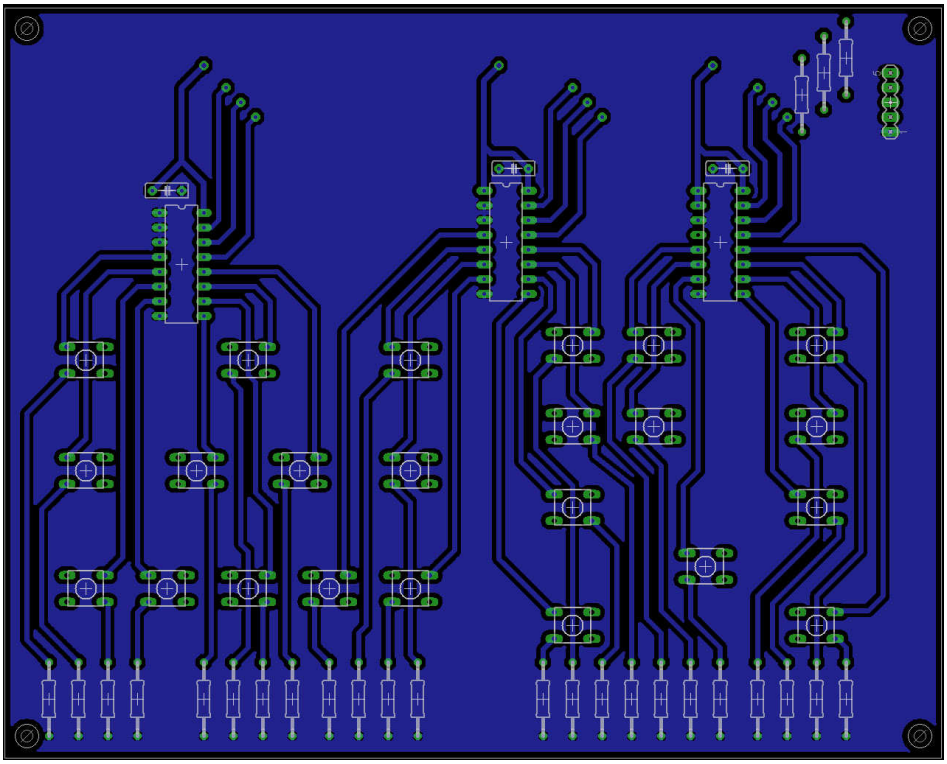


Fig. 3.4. Capa inferior de la placa de circuit imprès de la consola

3.2.2 Tauler Visualitzador

En aquest cas la complexitat de l'entramat de línies ha fet que sigui indispensable utilitzar les capes superior i inferior per dibuixar-hi pistes. Els *pads* (punts de connexió entre les potes dels dispositius i les pistes, en verd) i les vies (punts de connexió entre pistes que canvien de capa, també en verd) es repetiran a les dues cares i els seus forats estaran recoberts amb el mateix material conductor de les pistes. Això permetrà també que només calgui soldar per la capa inferior.

A la part superior dreta s'hi han marcat dos forats per cargolar la botzina i a la part superior esquerra es poden veure, com en el cas de la consola, els sòcols per connectar les línies necessàries a la placa d'entrenament, on hi ha el microcontrolador.

Com en el cas de la placa de la consola, s'han evitat els angles de 90º en el dibuix de les pistes per una correcta circulació del corrent.

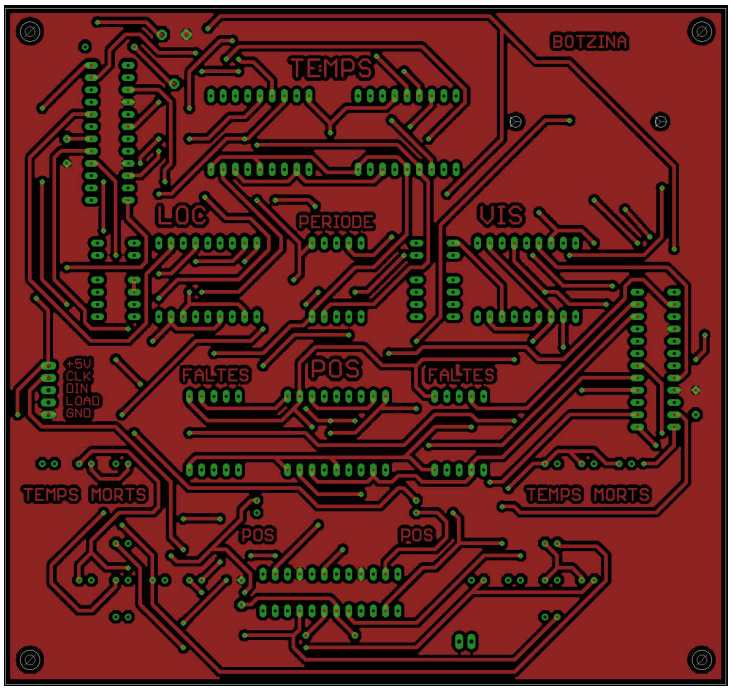


Fig. 3.5. Capa superior de la placa de circuit imprès del tauler visualitzador

En ambdós casos s'ha metal·litzat l'espai sense pistes per estalviar feina a la fresadora que fabrica les plaques. S'ha aprofitat per associar aquests espais a massa quan ha estat possible.

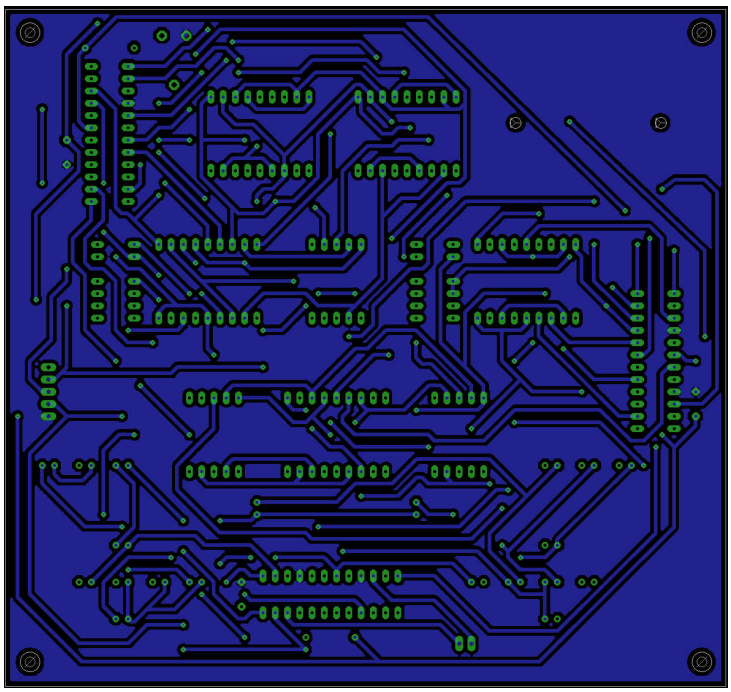


Fig. 3.6. Capa inferior de la placa de circuit imprès del tauler visualitzador

3.3 Diagrama de flux

De manera similar al cas de les verificacions, el programa principal comença definint les variables. Seguidament inicialitza la pantalla LCD i els *displays* de set segments del tauler, imprimint el valor de les variables a controlar. Per últim habilita tant les interrupcions globals com les GPIO i les dels dos temporitzadors necessaris pels dos comptes, el de temps de període i el de possessió.

Per inicialitzar el tauler se'l fa entrar en mode de prova (pins de segment dels MAX7219 a nivell alt, tot el tauler s'encén) durant breus instants per comprovar que tots els digits, els leds i la botzina funcionen.

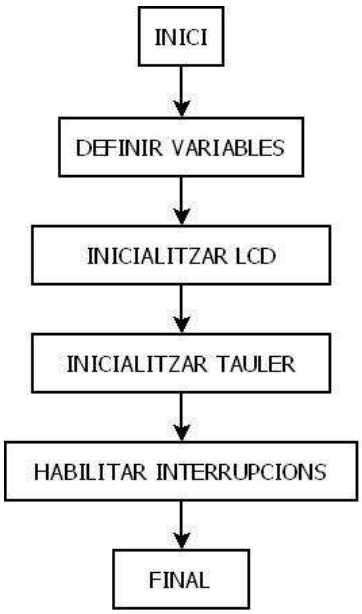


Fig. 3.7. Programa principal

A partir d'aquest moment entren en joc les interrupcions que produeixen els polsadors i els temporitzadors.

El diagrama de flux de les interrupcions GPIO es troba a l'annex 3 degut al seu tamany. En ell es poden veure les reaccions que comporta prémer cada un dels 23 polsadors del teclat de la consola.

El temps de període està regit pel Timer16_1 (figura 3.8). Inicialment els polsos del rellotge del temporitzador tenen una freqüència d'1 Hz fins que el compte enrera arriba al minut. Llavors la freqüència del rellotge augmenta fins als 100 Hz per portar el compte en centèsimes de segon. El led de doble dígit que mostrava els minuts ara ha de mostrar els segons i el que mostrava els segons ha de mostrar les centèsimes.

En cas que es vulgui incrementar el temps que queda de període i passi de ser de menys d'un minut a més d'un minut, es tornaran a mostrar minuts i segons. En el cas contrari, si es resta el compte fins que es deixa a menys d'un minut, es manté la coherència, es mostren segons i centèsimes.

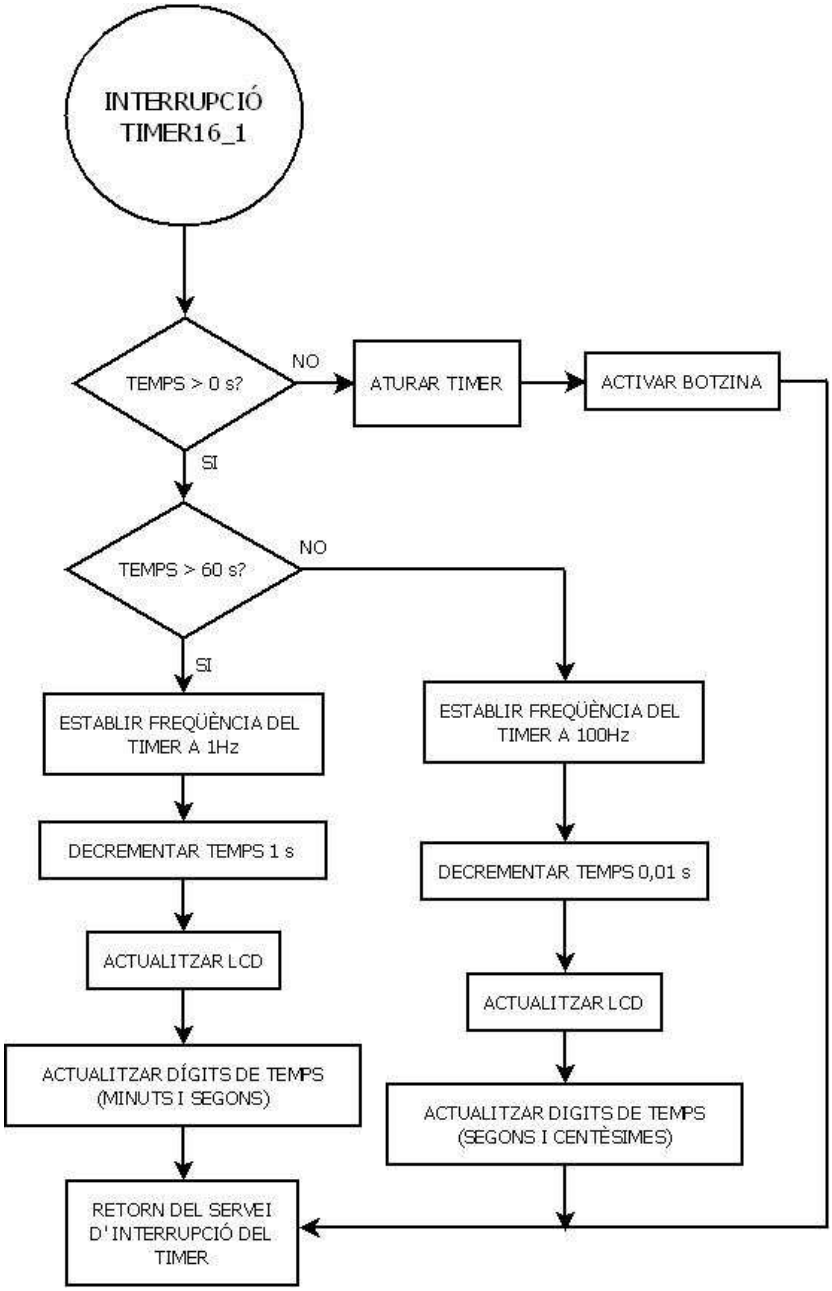


Fig. 3.8. Interrupció del Timer16_1

El temps de possessió sempre es mostra en segons, per tant la freqüència del rellotge del temporitzador que el controla (el Timer16_2) és d'1 Hz. La seva particularitat és que s'ha de deixar de mostrar en el moment en que sigui superior al temps de període.

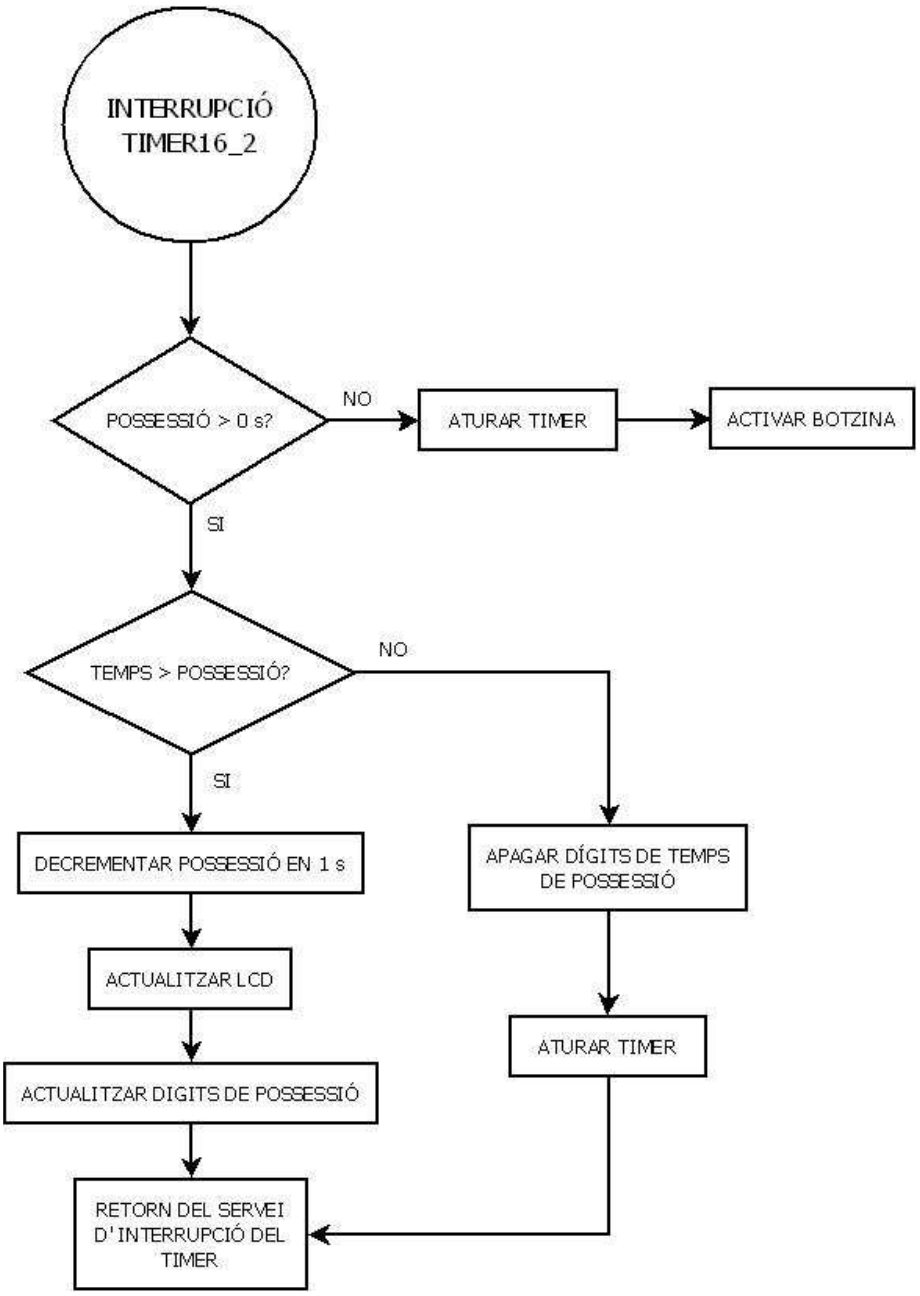


Fig. 3.9. Interrupció del Timer16_2

3.4 Verificació final

Aquest és el muntatge final després de construir les plaques i soldar en elles tots els components:

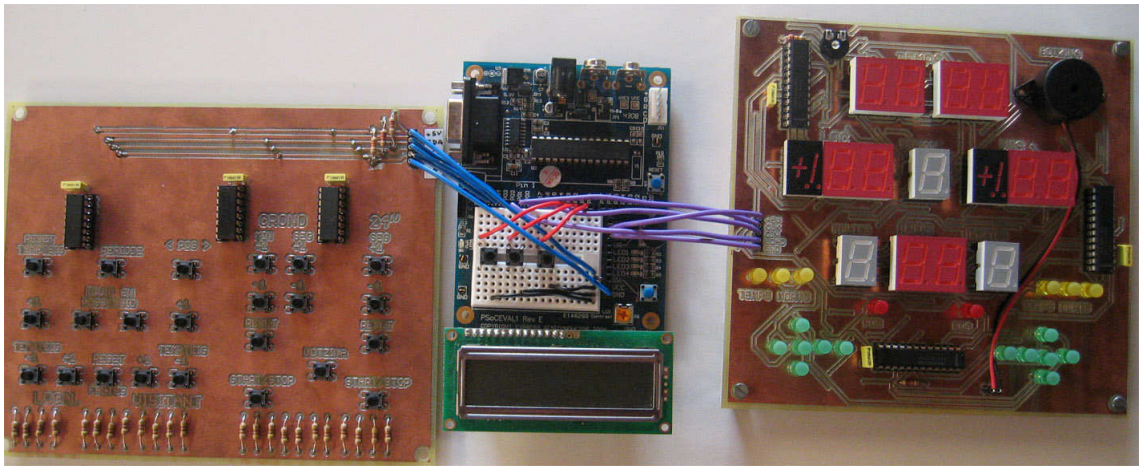


Fig. 3.10. Prototip de marcador de bàsquet final

La placa de l'esquerra és la consola, la del mig es la PSoCEval1, la qual aprofitem per mostrar dades per la pantalla LCD i per ubicar les tres tecles indicadores dels temps morts locals i visitants, i la de la dreta es el tauler de visualització.

A través de PSoC Designer s'elabora el programa final i s'envia al microcontrolador. El programa ha de permetre el funcionament del marcador respectant els requeriments inicials. Per comprovar-ho es fa una petita simulació , seguint els esdeveniments d'un partit de bàsquet hipotètic.

Un cop alimentat el marcador s'entra en una fase d'uns dos segons en què tots els leds (la botzina també) s'encenen. Un cop passat aquest temps de comprovació es pot utilitzar el marcador de manera normal i es procedeix a la simulació.

Primerament es fixa el període de joc. El següent pas és activar el temps de partit quan un dels jugadors toca la pilota en el salt inicial. Quan un dels equips pren possessió de la pilota s'activa el compte enrera dels 24 segons, s'encén la fletxa que indica el sentit d'atac de l'equip que no ha aconseguit la pilota i s'encén el led vermell de possessió actual la pilota.

A partir d'aquí es comproven les tecles de suma i resta de tempteigs, de faltes, de minuts i segons dels temps, els leds indicadors de temps morts i la botzina.

A la figura 3.11 (pàgina següent) es pot veure una imatge de la simulació.

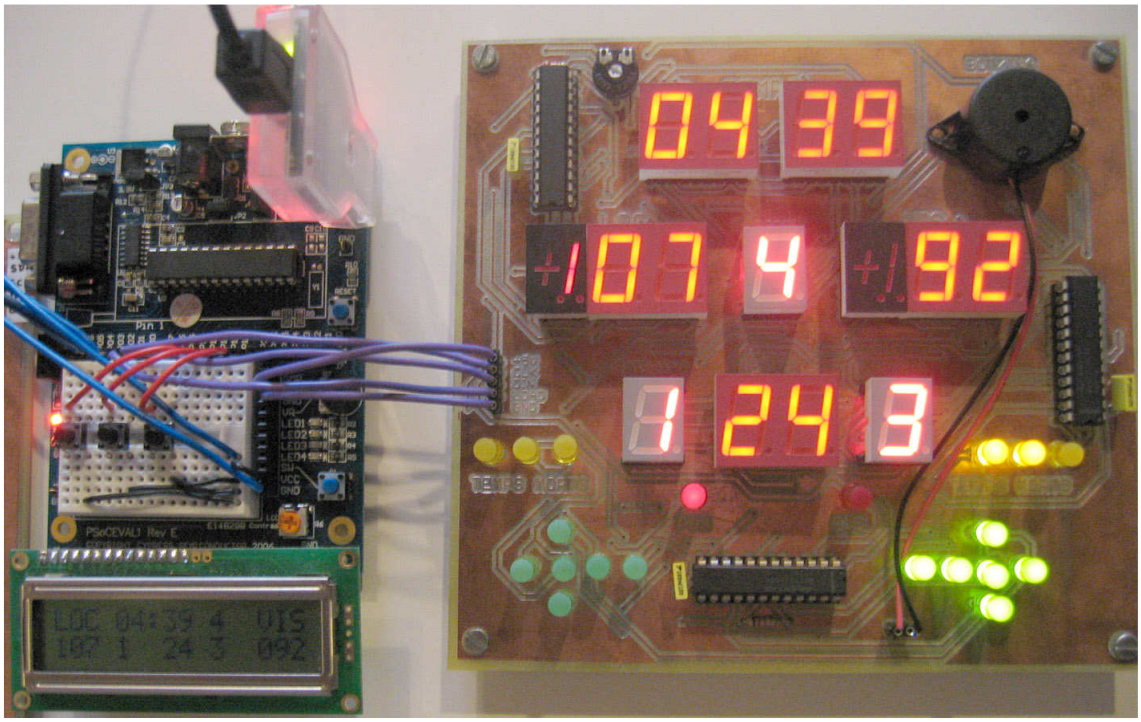


Fig. 3.11. Comprovació final

Després d'un temps de simulació suficient i de comprovar totes les tecles i les possibilitats del marcador es conclou que el prototip de marcador de bàsquet d'aquest projecte compleix els requeriments proposats a l'inici del treball i que per tant és capaç de seguir correctament qualsevol esdeveniment d'un partit de bàsquet oficial.

3.5 Aspectes ambientals

Els components emprats en aquest projecte compleixen la directiva RoHS, incloent la placa d'entrenament per PSoC.

3.5.1 Directiva RoHS

La directiva RoHS (Restriction of Harzadous Substances) és la directiva 1050/95/CE de restricció de certes substàncies perilloses en aparells elèctrics i electrònics. Va ser adoptada el febrer de 2003 per la Unió Europea i va entrar en vigor l'1 de juliol de 2006.

Restringeix l'ús de 6 substàncies perillosos en la fabricació de diversos tipus d'equips elèctrics i electrònics. Les concentracions màximes d'aquestes substàncies respecte el pes en materials homogenis són del 0,1% en els casos del plom, del mercuri, del crom VI, del PBB i del PBDE (les dues últimes són

retardants de les flames que s'utilitzen en alguns plàstics) i del 0.01% en el cas del cadmi. Això significa que aquests límits no s'apliquen al pes del producte final, sinó al pes de cada tipus de material que forma el producte final i que pot ser separada mecànicament del mateix.

No contempla el material de les bateries ja que aquestes es regeixen per la seva pròpia directiva.

S'aplica als següents equips:

- Electrodomèstics petits i grans
- Equips de comunicació i IT
- Aparells elèctrics de consum
- Aparells d'enllumenat
- Eines elèctriques i electròniques
- Màquines expenedores
- Joguines, equips esportius i de temps lliure.

Per tant, degut a aquest últim punt, el marcador que s'ha construït en aquest projecte també hauria de complir aquesta directiva.

Estan exemptes de complir-la els aparells mèdics, els instruments de vigilància i control i els equips dissenyats amb propòsits militars o espacials. També estan exemptes una llista de casos que es troben a l'annex de la directiva.

En cas de no complir-se, la RoHS responsabilitza al productor de l'equip i s'aplica tant a productes fabricats a la Unió Europea com a productes importats. Lleis similars o de reciclatge són adoptades per països com Xina, Japó o per l'estat de Califòrnia, cosa que fa que aquest últim aspecte no sigui massa problemàtic i que el compliment de la RoHS sigui un tema de caràcter mundial.

Fig.3.X. Exemple de diferents distintius de compliment de la directiva

3.6 Conclusió

L'elaboració d'aquest projecte ha comportat una iniciació important en el món dels microcontroladors, especialment al PSoC de Cypress. S'ha comprès quins criteris tenir en compte per escollir el més adient, les seves possibilitats i la manera de connectar i programar els seus mòduls interns per obtenir els resultats desitjats.

També ha donat una idea de la quantitat de dispositius electrònics que existeixen al mercat destinats a solucionar les mancances d'altres dispositius, com en el cas de necessitar més ports al microcontrolador o la de controlar fàcilment una sèrie de leds o de *displays*.

Els sistemes de comunicació entre aquests dispositius, que han permès l'estalvi de ports i, per tant, d'expansors i de línies, també ha estat un aprenentatge important del qual no es tenia coneixement

Una altra experiència important adquirida ha estat la de dissenyar una placa de circuit imprès a partir d'un esquema i la seva posterior construcció.

Segurament la part més complicada del projecte ha estat l'organització de les tasques en el temps per complir els terminis. Els imprevistos, els errors de programació o de muntatge i dependre d'altres persones són factors que poden fer que una mala planificació del temps no permeti acabar el projecte a temps.

Tots aquests coneixements s'han anat adquirint intentant dissenyar i construir un aparell electrònic senzill i habitual en un àmbit conegut, fet que li ha aportat motivació. Aquesta ha anat augmentant a partir de les verificacions satisfactòries, moment en que s'ha començat a veure que els passos que s'havien seguit eren els correctes i que la seva construcció era possible. Abans, però, han estat necessàries hores de recerca infructuosa, proves fallides i d'una certa frustració. Finalment, quan s'aconsegueix l'objectiu i el marcador funciona correctament, s'obté la sensació de que tot ha valgut la pena.

CAPÍTOL 4. Bibliografia i línies futures de treball.

4.1 Comunicació sense fils

Una línia d'investigació futura clara per a aquest projecte és la de comunicar consola i tauler sense cables. Seria una gran oportunitat per posar en pràctica els coneixements adquirits en assignatures que tracten els sistemes i les tecnologies de radiofreqüència i els sistemes de transmissió digital.

4.2 Tempteig automàtic

Una possibilitat interessant per a un marcador de bàsquet és convertir en automàtiques alguna de les seves funcions, com ara la d'acumular el tempteig. Es tracta de dotar les cistelles amb un mecanisme electrònic que detecti el pas d'una pilota per l'anella i que enviï aquesta informació cap al marcador.

Aquesta característica no seria de gaire ajuda en un partit oficial ja que el sensor que detecta el pas de la pilota no sap si la pilota prové d'un llançament d'1, 2 o 3 punts, a menys que es complementi amb un sistema similar a l'ull de falcó (*Hawk-Eye*) que actualment s'utilitza en tennis i en criquet. Aquest sistema de seguiment de la trajectòria de la pilota a més hauria de detectar la zona del terreny de joc que el llançador trepitja just abans del llançament i hauria de ser pràcticament instantani. Aleshores es podria prescindir de l'auxiliar de taula que en partits d'alt nivell es dedica només a actualitzar el tempteig del partit al marcador.

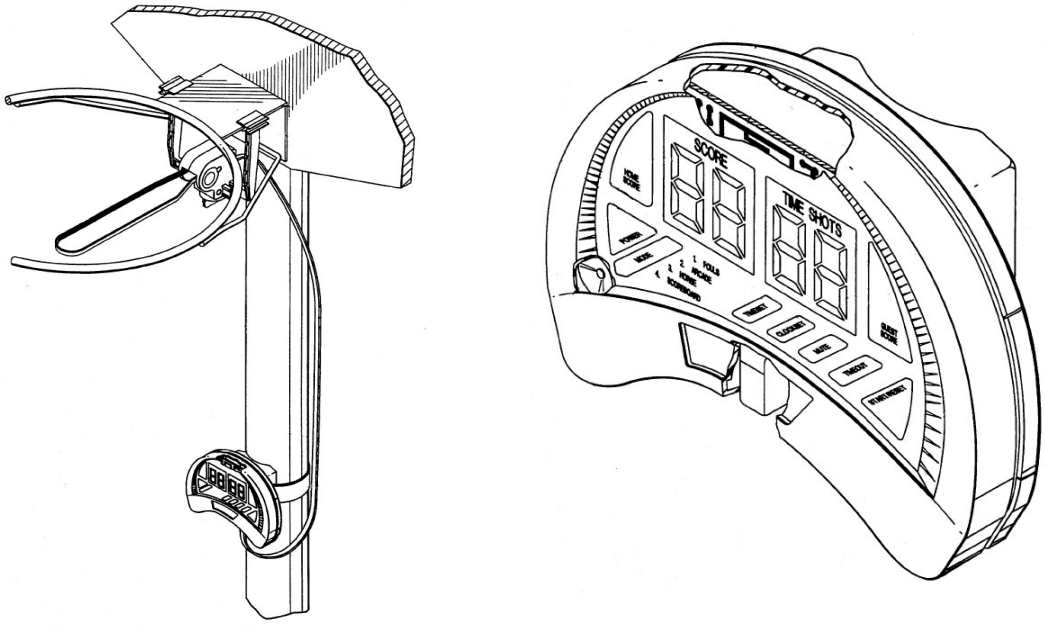


Fig. 4.1. Sistema de tempteig automàtic per palanca

Sí seria útil en casos en què no fes falta distingir el valor del llançament, com per exemple en un entrenament de llançament. Aleshores podria portar el compte de les cistelles aconseguides pel llançador. Si a més el mecanisme incorporés un sistema de detecció de vibració, es podrien comptar el nombre total de llançaments, encertats i fallats, sempre que toquin l'anella o el tauler o passin per l'anella sense tocar-la. Això permetria donar la informació del percentatge d'encert. També caldria tenir en compte que alguns llançaments poden tocar diverses vegades la cistella i fer que fos comptabilitzat només un cop.

4.3 Bibliografia

[1] Castillo Jordán, A., *Diseño, simulación e implementación de un embedded system basado en PSoC de Cypress: aplicación a la síntesis de sonido*, 2008.

[2] Cypress Semiconductor Corporation, *Technical Reference Manual (TRM)*, Document nº 001-14463 Rev. *C, San José, California (USA).

[3] Ashby, R., *Designer's Guide to the Cypress PSoC*, ed. Elsevier, Burlington, Massachusetts (USA), 2005.

[4] Peña Basurto, M.A. i Cela Espín J.M., *Introducción a la programación en C*, Edicions UPC, Barcelona, 2000.

[5] Diari Oficial de la Unió Europea, "DIRECTIVA 2002/95/CE DEL PARLAMENTO EUROPEO Y DEL CONSEJO de 27 de enero de 2003 sobre restricciones a la utilización de determinadas sustancias peligrosas en aparatos eléctricos y electrónicos".

[6] Pàgina web de Cypress Semiconductor, http://www.cypress.com.

[7] Pàgina web de CadSoft, http://www.cadsoftusa.com.

[8] Pàgina web de Maxim Integrated Products, Inc., http://www.maxim-ic.com.

[9] Pàgina web de Baybor S.A., http://www.baybor.com

[10] Pàgina web de Mondo, http://www.mondoworldwide.com/index_eu_es.cfm

[11] Pàgina web de la Federació Catalana de Basquetbol, http://www.basquetcatala.cat

[12] Cercador de fulls d'especificacions de components electrònics, http://www.datasheetcatalog.com

[13] http://www.wikipedia.org

[14] Bus I2C, http://robots-argentina.com.ar/Comunicacion_busI2C.htm

[15] Bus SPI, http://www.mct.net/faq/spi.html

[16] I2C vs. SPI, http://embedded.com/columns/beginerscorner/9900483?_requestid=525017

[17] Fòrum d'Arduino, http://www.arduino.cc/cgi-bin/yabb2/YaBB.pl?num=1258006748

[18] Pàgina web d'Onda Radio S.A., http://www.ondaradio.es

4.4 Annexos

4.4.1 Annex 1: tutorial per PSoC Designer 5

En aquest tutorial es mostren acuradament els passos seguits per programar el microcontrolador del marcador de bàsquet mitjançant el programa de lliure distribució PSoC Designer 5. S'espera que a partir d'aquest exemple concret el lector es faci una idea del funcionament d'aquesta nova versió del programa i que la pugui aplicar a les seves necessitats.

La primera pantalla que es mostra en obrir el programa és la següent:

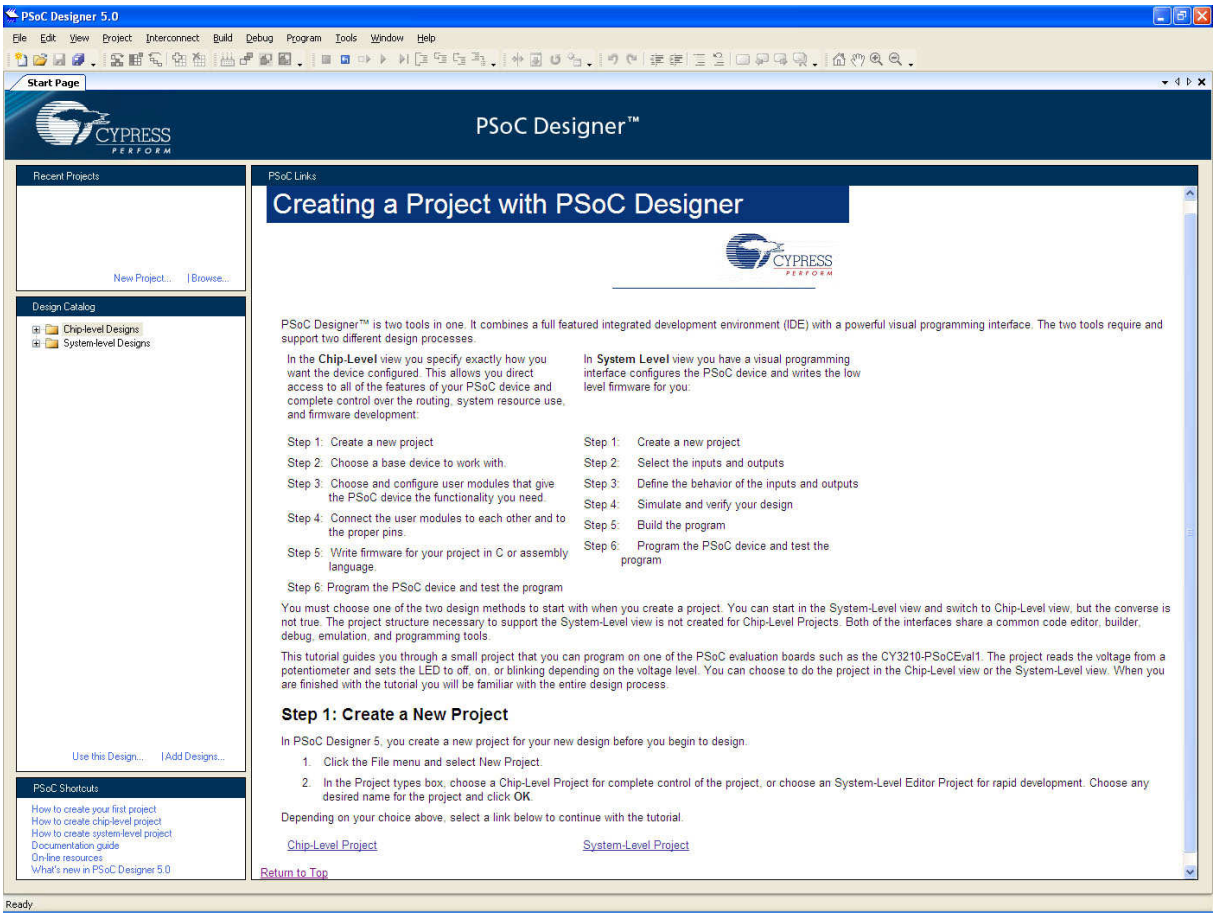


Fig. 4.2. Pantalla de presentació del programa

on es pot trobar un tutorial que explica les diferències entre el *Chip-Level Project* i el *System-Level Project*, resumides a l'apartat 2.2.3 d'aquest projecte.

Per començar un nou projecte hi ha diverses opcions, pitjar sobre ***New Project...*** a la finestra de *Recent Projects* de la part superior esquerra de la pantalla inicial (figura 4.3), desplegar el menú ***File*** i pitjar sobre ***New Project...*** (figura 4.4) o bé utilitzar el mètode abreujat, pitjant les tecles ***Ctrl***, ***Shift*** i ***N***.

Si es vol obrir un projecte ja creat recentment apareixerà a la finestra *Recent Projects*.

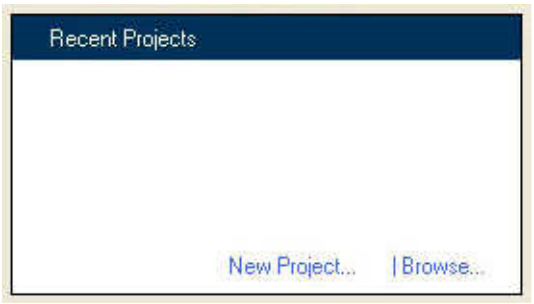


Fig. 4.3. Finestra *Recent Projects*

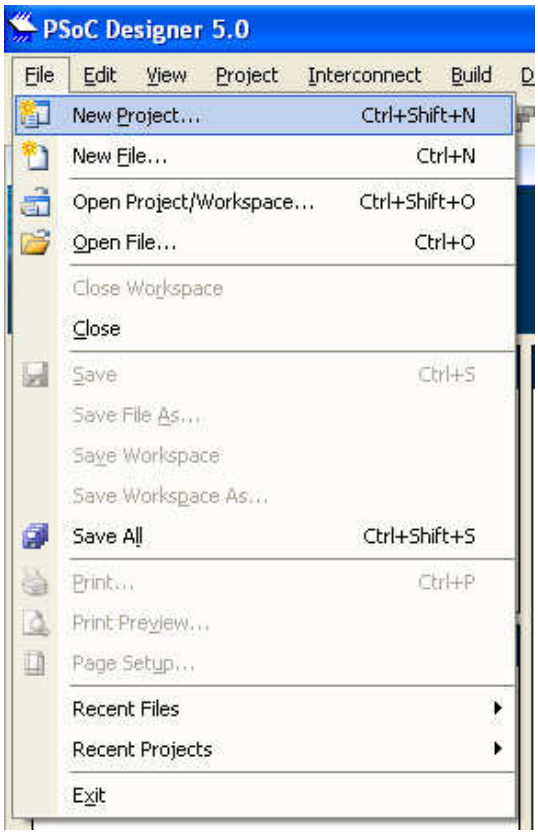


Fig. 4.4. Menú *File*

Si s'està creant un nou projecte, emergeix la següent finestra:

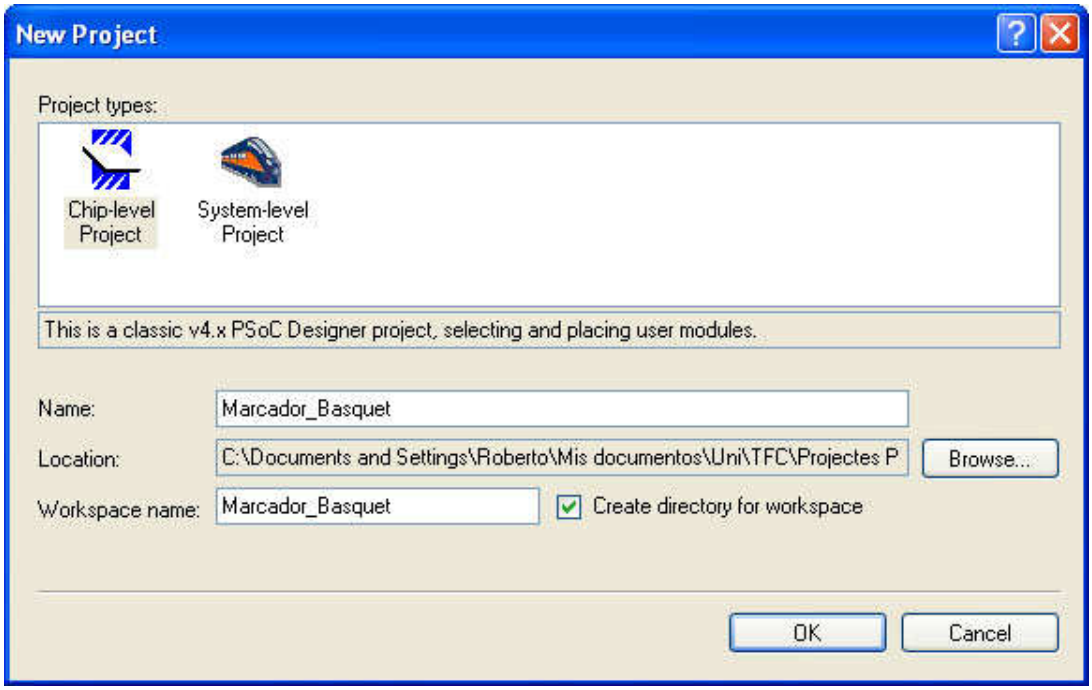


Fig. 4.5. Finestra *New Project*

on es pot escollir entre els dos tipus de projecte i donar-li nom i ubicació dins l'ordinador. En aquest cas s'escull el *Chip-Level Project*, ja que s'ha volgut utilitzar llenguatge de programació. Un cop realitzades aquestes operacions es pitja ***OK***.

Seguidament es demana el tipus de PSoC a programar i el llenguatge que es vol fer servir:

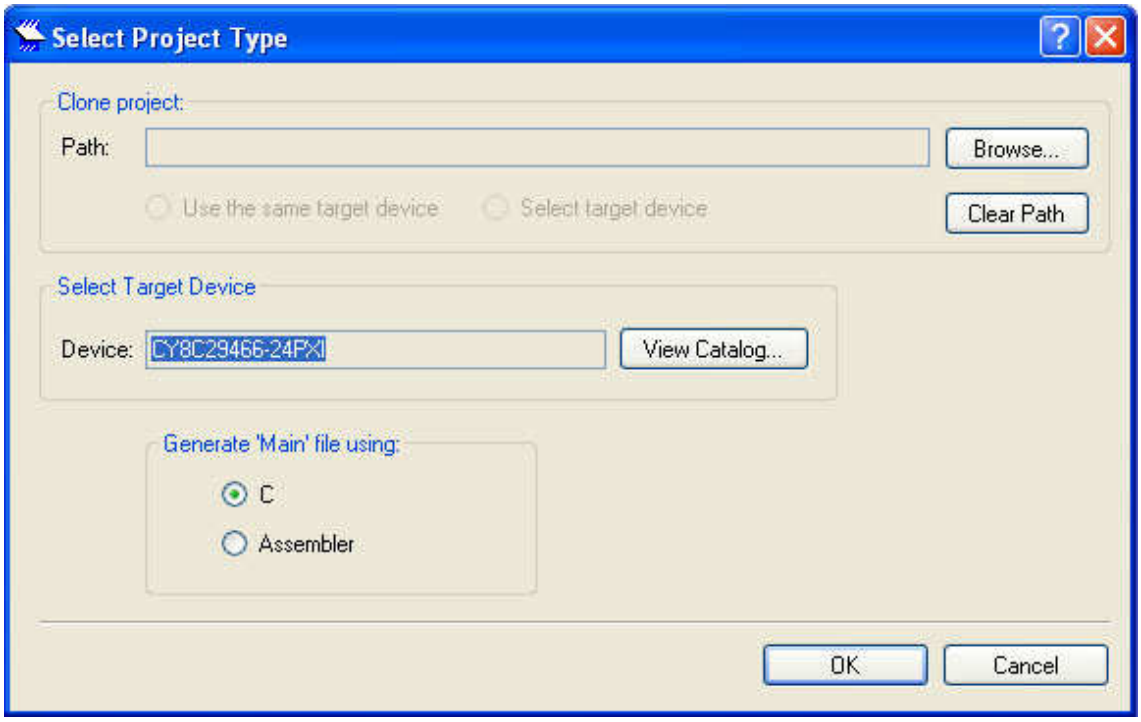


Fig. 4.6. Finestra d'elecció de microcontrolador i llenguatge

En aquest cas, com ja s'ha vist anteriorment, s'ha programat en llenguatge C i el microcontrolador utilitzat és el CY829466-24PXI, el qual es selecciona a la pantalla que apareix pitjant ***View Catalog...*** (figura 4.7).

En aquesta pantalla apareix un catàleg de tots els tipus de PSoC de Cypress que existeixen i permet la seva ordenació segons diversos paràmetres com el nombre de blocs, la capacitat de les memòries Flash i RAM, la tensió d'alimentació, el tipus d'encapsulat, etc., que facilita trobar el microcontrolador concret.

Un cop trobat el dispositiu, es pitja ***Select*** i es retorna a la finestra de la figura 4.6.

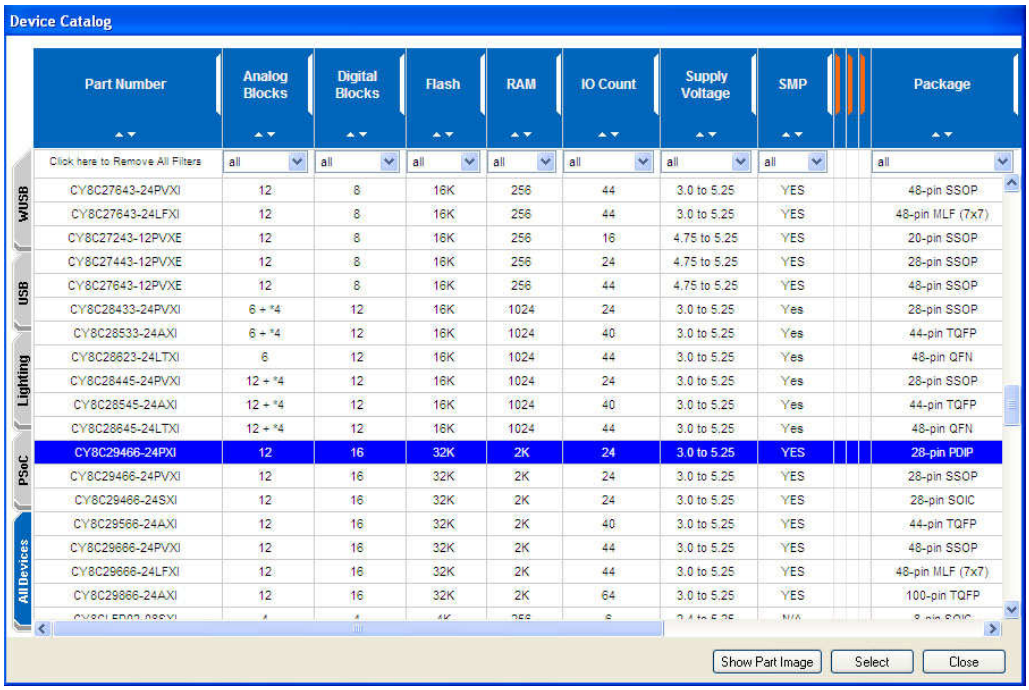


Fig. 4.7. Catàleg dels PSoC de Cypress

Un cop omplerta la finestra de la figura 4.6, es pitja ***OK*** i es passa a la pantalla principal del projecte (figura 4.8), on comença la programació i la connexió dels dispositius interns necessaris del microcontrolador.

La pantalla principal està dividida en 6 finestres:

- *Global Resources*, on es troba el llistat de recursos globals com la tensió d'alimentació, la freqüència del rellotge intern del microcontrolador i de les altres fonts de rellotge derivades (VC1, VC2 i VC3), la possibilitat d'utilitzar un rellotge extern, etc. Sota d'aquesta finestra hi ha una altra on apareix informació sobre el recurs seleccionat com els valors que admet o a quins registres afecta.

- *Properties*, on apareixen les propietats del dispositiu intern col·locat als blocs i seleccionat. Algunes d'elles es poden modificar segons les necessitats. Sota aquesta finestra també hi ha una altra finestra que dóna informació sobre el paràmetre seleccionat.

- *Pinout*, on es pot determinar el comportament dels pins, el seu nom, connectar-los a les línies internes o habilitar una interrupció quan rep certa informació.

- *Workspace Explorer*, que permet navegar entre les carpetes i els arxius del projecte creat. S'hi poden trobar els dispositius afegits, les seves llibreries, el fitxer *main.c* i altres fitxers que serà necessari modificar per fer funcionar les interrupcions dels dispositius.

- *User Modules*, conté un llistat agrupat en categories de tots els dispositius interns de què disposa el microcontrolador. Algunes d'aquestes categories són amplificadors, conversors analògic-digital i digital-analògic, comptadors, temporitzadors, comunicacions analògiques i digitals, filtres, etc.

- Finestra principal, on es mostra un diagrama dels blocs analògics i digitals interns del PSoC, així com línies de connexió i els pins d'entrada/sortida. El programa permet fer-li zoom i moure's per ella i realitzar connexions i altres accions pitjant sobre els elements.

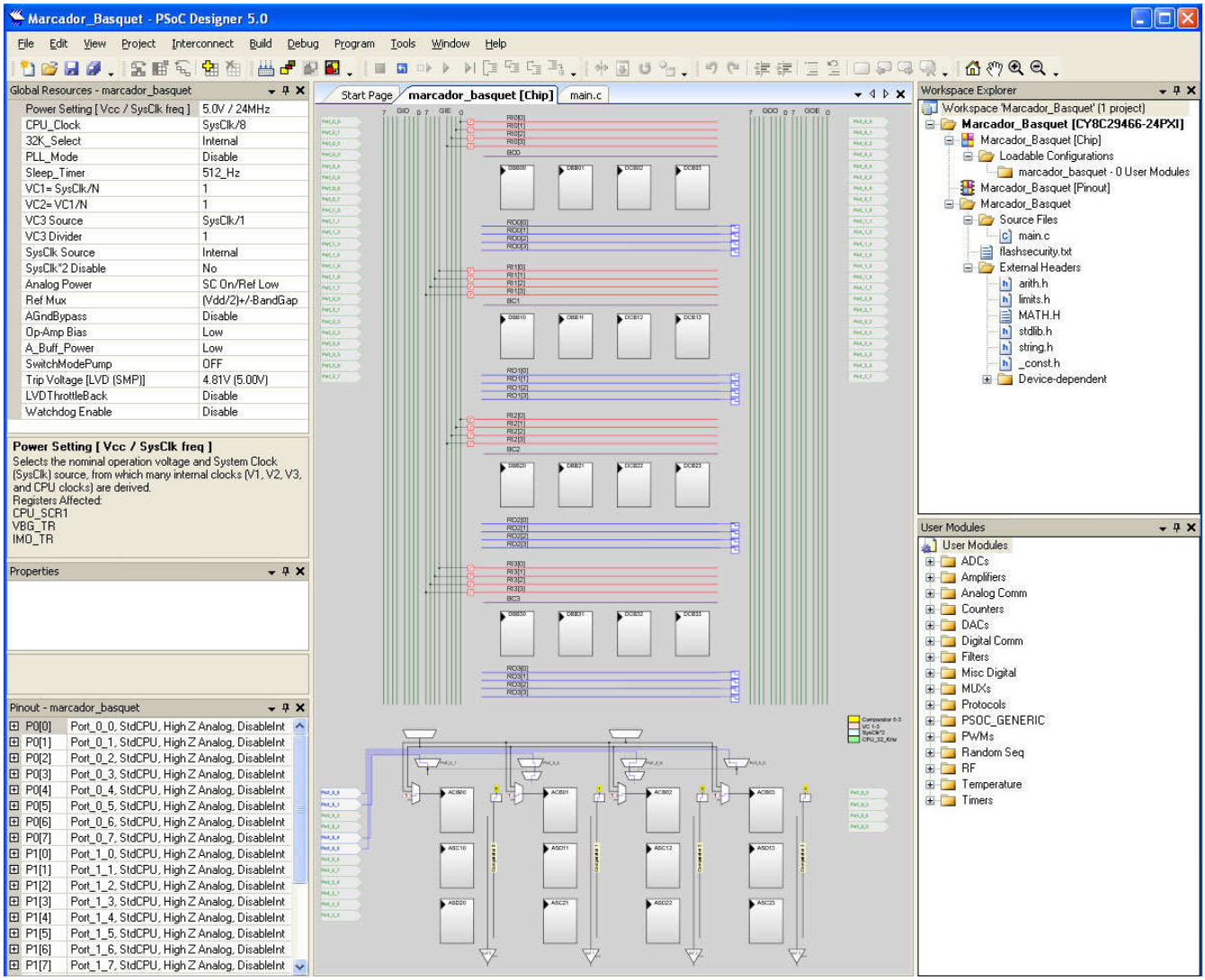


Fig. 4.8. Pantalla principal del projecte

Abans de començar a escollir i programar dispositius s'habilita el mode gratuït del compilador de llenguatge C (si no s'ha fet durant la instal·lació del programa o en projectes anteriors). Per fer-ho cal desplegar el menú ***Project***, pitjar en ***Settings...***, escollir el compilador HI-TECH i pitjar ***Switch to Lite Mode***. A la finestra *Options* apareixerà escrit *--mode=lite*, que vol dir que ha estat activat correctament.

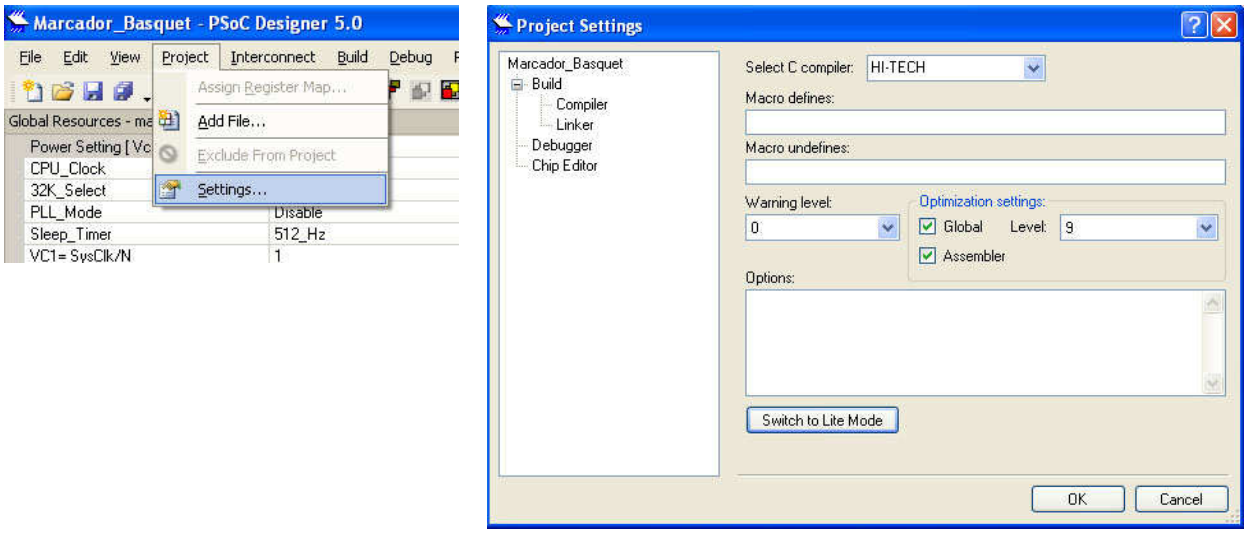


Fig. 4.9. Com accedir a la finestra *Project Settings*

Aquest mode és menys complet que el de pagament però satisfà les necessitats del projecte. També hi ha la possibilitat d'activar el *PRO mode* durant un període d'avaluació de 45 dies buscant el programa ***HI-TECH Software*** instal·lat automàticament a l'ordinador (després de la instal·lació del PSoC Designer 5) i pitjant ***Activate or uninstall***.

Per aprofitar on ens trobem s'activa el control de generació d'interrupcions (el qual permetrà treballar amb elles) pitjant en ***Chip Editor*** del menú de l'esquerra i activant la casella *Enable interrupt generation control*. Es prem ***OK*** i es torna a la pantalla principal.

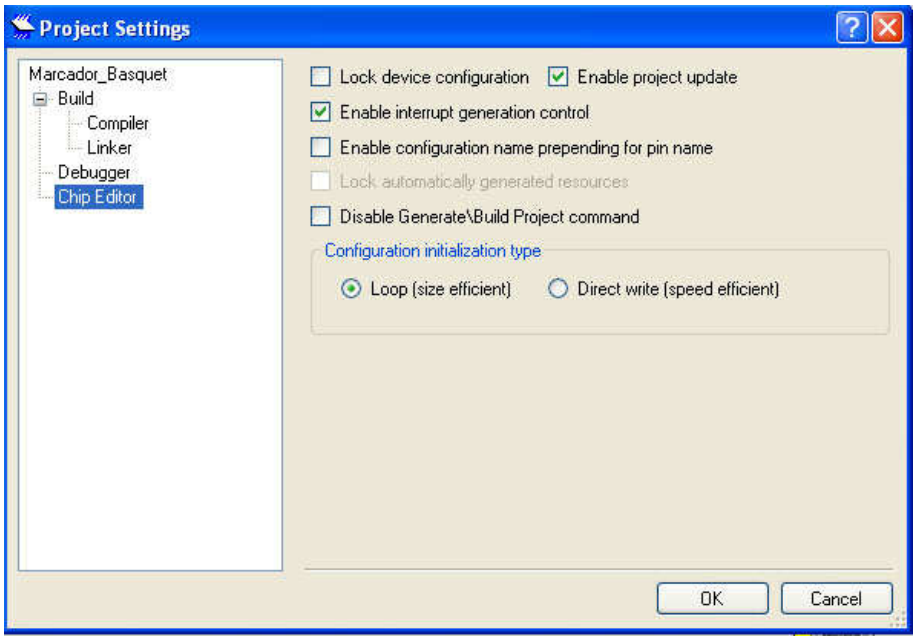


Fig. 4.10. Finestra *Chip Editor* on activar les interrupcions

Abans de començar a programar s'explicarà com afegir dispositius al projecte. El primer dispositiu a afegir pot ser la pantalla LCD, que permet practicar amb ella per agafar-li confiança al programa. Per afegir qualsevol dispositiu es pot anar a la finestra *User Modules*, desplegar les carpetes fins trobar-lo, pitjar a sobre amb el botó esquerra del ratolí i pitjar la opció *Place*.

Abans, però, és convenient donar un cop d'ull al seu *Datasheet* (full d'especificacions) per comprovar que un cop afegit i programat satisfarà els requeriments inicials. La figura 4.11 també mostra com trobar el *Datasheet*.

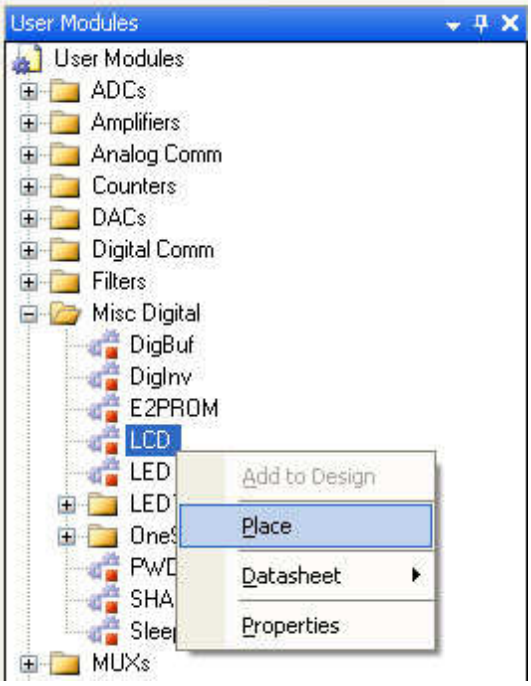


Fig. 4.11. Afegir un dispositiu

Tots els Dataheets mantenen la mateixa estructura. Primer presenten una taula amb les famílies de PSoC que disposen del dispositiu en qüestió, la quantitat de blocs analògics i digitals que ocupen, la memòria Flash i RAM que consumeixen i la quantitat de pins que necessiten. A continuació es mostra un breu resum de les capacitats del dispositiu i una imatge esquemàtica del mateix i seguidament una sèrie d'apartats que aprofundeixen en el seu funcionament, els paràmetres i recursos susceptibles de ser escollits a la finestra *Properties*, les rutines per programar-lo (*API's*) i un exemple de programa senzill, en llenguatge C i en assemblador.

La programació s'ha d'anar alternant amb la inclusió de dispositius d'una manera ordenada i lògica. En el cas del marcador de bàsquet els processos són pràcticament paral·lels i aquesta ordenació no és important, però resulta d'ajuda tenir la pantalla LCD preparada per veure els valors de les variables, sobretot quan es treballa amb la versió *lite*, que no permet emprar l'eina *debugger*. Aquesta eina, pròpia de la versió *PRO*, permet executar el programa creat pas per pas mentre s'observa com van canviant les variables. Resulta molt útil per trobar i corregir errors quan el redactat és correcte però no s'aconsegueixen els resultats previstos.

En total, per recrear el marcador de bàsquet s'han fet servir 5 dispositius:

LCD, que mostrarà un resum dels paràmetres que es poden veure al tauler electrònic. No ocupa cap bloc intern.

Timer8, temporitzador de 8 bits, que a una freqüència de 100 Hz permet recrear les centèsimes de segon. Ocupa un bloc digital.

Timer16, temporitzador de 16 bits, necessari per aconseguir una freqüència d'1 Hz per recrear els segons. Ocupa dos blocs digitals.

I2Cm, permet la comunicació digital I2C amb els expansors de ports als quals estan connectats tots els polsadors de la consola. No ocupa cap bloc.

SPI, permet la comunicació digital SPI amb els controladors de leds. Ocupa un bloc digital.

El redactat en llenguatge C es fa en el fitxer *main.c* al qual es pot accedir des de la finestra *Workspace Explorer*. Està ubicat dins de la carpeta *Source Files* que està dins de la carpeta del projecte en qüestió, en aquest cas la carpeta *Marcador_Basquet*.

Obrint el fitxer .c apareix la següent pantalla:

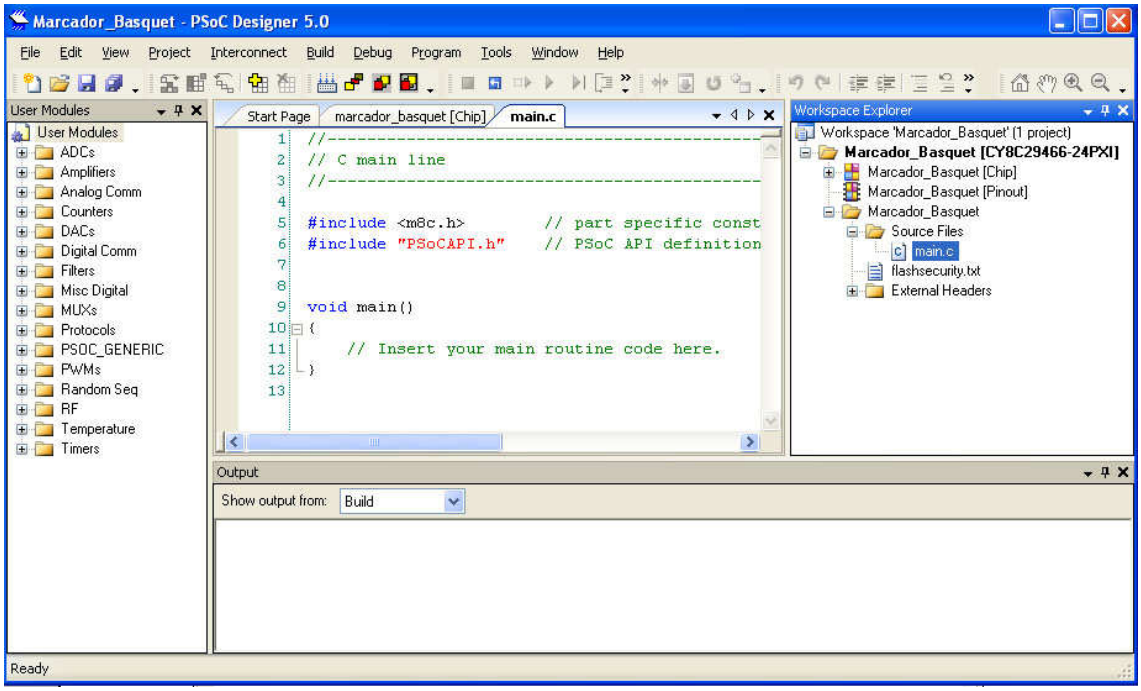


Fig. 4.12. Pantalla del fitxer *main.c*

on tornem a trobar les finestres User Modules i Workspace Explorer, a més d'una de nova, Output, on es podran veure els resultats de les recerques d'errors que es vagin fent. El redactat es farà a la finestra principal.

4.4.2 Annex 2: codi del programa

```
//--------------------------------------------------------------------------
// MARCADOR ELECTRÒNIC DE BÀSQUET - Robert Pinedo
//--------------------------------------------------------------------------

#include <m8c.h>                    // Part specific constants and macros
#include "PSoCAPI.h"                // PSoC API definitions for all User Modules
#include "Timer16_1.h"              // Timer per la possessió
#include "Timer16_2.h"              // Timer pel temps de partit

// Funcions interrupció

#pragma interrupt_handler PremTecla             // Interrupció GPIO pels polsadors
#pragma interrupt_handler CadaSegon             // Interrupció del Timer16_1 per la possessió
#pragma interrupt_handler CadaSegonCentesima    // Interrupció del Timer16_2 pel temps

// Accès als ports d'Entrada/Sortida

#define Bit(bitNumber)(1<<(bitNumber))

// Comprovació de l'estat dels polsadors mitjançant operació AND

#define PolsadorTMV (PRT1DR&Bit(1))         // Polsadors de temps mort
#define PolsadorTMR (PRT1DR&Bit(2))
#define PolsadorTML (PRT1DR&Bit(3))
#define PCFint    (PRT0DR&Bit(5))           // Senyal d'interrupció dels expansors de ports

// Taula de direccions de registres del MAX7219

#define NO_OP  0x00           //     Emprat en muntatges en cascada
#define DIG0   0x01           //     Accedeix al primer dígit del marcador
#define DIG1   0x02           //     Accedeix al segon dígit del marcador
#define DIG2   0x03
#define DIG3   0x04
#define DIG4   0x05
#define DIG5   0x06
#define DIG6   0x07
#define DIG7   0x08
#define DECODE        0x09          // Estableix quins dígits es decodifiquen en BCD i quins no
#define INTENSITY     0x0a          // Modifica la brillantor dels leds
#define SCAN_LIMIT    0x0b          // Estableix quants dígits seran utilitzats
#define SHUTDOWN      0x0c          // Entra o surt del mode shutdown (tot apagat).
#define DISPLAY_TEST  0x0f          // Entra o surt del Display Test mode (tot encès).

//Puls LOAD dels MAX7219

#define LOAD_0 (PRT0DR&=~Bit(2))    // Estableix el senyal LOAD a 0
#define LOAD_1 (PRT0DR|=Bit(2))     // Estableix el senyal LOAD a 1

// Variables globals

BYTE TanteoLoc=0x00, TanteoVis=0x00;                // Variables de tempteig acumulat
BYTE pos=24, minut=10, segon=0, centesima=0;        // Temps de possessió i de període
BYTE FaltesLoc=0, FaltesVis=0;                      // Faltes personals acumulades
BYTE periode=0;
BYTE TmortL=0, TmortV=0;

BYTE unitat, desena;
BYTE PosDesena, PosUnitat;
int aux1=0, aux2=0, AuxSentit=0;

char Cunitat[2], Cdesena[2];                // La pantalla LCD només imprimeix "null terminated
strings"
char CFaltesLoc[2], CFaltesVis[2];
char Cperiode[2];
char CPosDesena[2], CPosUnitat[2];

// Llistat de funcions

void PremTecla();                           // Funció interrupció GPIO
void CadaSegon();                           // Funció interrupció Timer16_1
void CadaSegonCentesima();                  // Funció interrupció Timer16_2
void Inicialitza_LCD();                     // Funció per inicialitzar la pantalla LCD
```

```
void Inicialitza_MAX7219();                 // Funció per inicialitzar el tauler de leds
void Envia_MAX7219_1(BYTE registre, BYTE dades);    // Envia dades al primer led driver
void Envia_MAX7219_2(BYTE registre, BYTE dades);    // Envia dades al segon led driver
void Envia_MAX7219_3(BYTE registre, BYTE dades);    // Envia dades al tercer led driver
void Prepara_Temps(BYTE enter1);                    // Separa minuts i segons en dècimes i unitats
void Prepara_Possessio(BYTE enter2);                // Separa possessió en dècimes i segons
void Botzina1();                                    // Emet el so de la botzina del temps
void Botzina2();                                    // Emet el so de la botzina de la possessió

// Funcions

void CadaSegonCentesima()
{
	if(minut>0)
	{
		if(segon==0)
		{
			minut--;
			segon=60;

			Prepara_Temps(minut);
			Envia_MAX7219_1(DIG1, unitat);
			Envia_MAX7219_1(DIG0, desena);

			Cunitat[0]=(char)(48+unitat);
			Cdesena[0]=(char)(48+desena);

			LCD_1_Position(0,4);
			LCD_1_PrString(Cdesena);
			LCD_1_Position(0,5);
			LCD_1_PrString(Cunitat);
		}
		segon--;

		Prepara_Temps(segon);
		Envia_MAX7219_1(DIG3, unitat);
		Envia_MAX7219_1(DIG2, desena);

		Cunitat[0]=(char)(48+unitat);
		Cdesena[0]=(char)(48+desena);

		LCD_1_Position(0,7);
		LCD_1_PrString(Cdesena);
		LCD_1_Position(0,8);
		LCD_1_PrString(Cunitat);
	}
	else if(minut<1)
	{
		Timer16_2_WritePeriod(0x05);

		if(centesima==0)
		{
			segon--;
			centesima=100;

			Prepara_Temps(segon);
			Envia_MAX7219_1(DIG1, unitat);
			Envia_MAX7219_1(DIG0, desena);

			Cunitat[0]=(char)(48+unitat);
			Cdesena[0]=(char)(48+desena);

			LCD_1_Position(0,4);
			LCD_1_PrString(Cdesena);
			LCD_1_Position(0,5);
			LCD_1_PrString(Cunitat);
		}
		centesima--;

		Prepara_Temps(centesima);
		Envia_MAX7219_1(DIG3, unitat);
		Envia_MAX7219_1(DIG2, desena);

		Cunitat[0]=(char)(48+unitat);
		Cdesena[0]=(char)(48+desena);

		LCD_1_Position(0,7);
```

```
                LCD_1_PrString(Cdesena);
                LCD_1_Position(0,8);
                LCD_1_PrString(Cunitat);
        }
        else if(minut==0&&segon==0&&centesima==0)
        {
                Timer16_2_Stop();
                Botzina1();
        }
        I2Cm_1_Start();
}

void CadaSegon()
{
        if(pos>0)
        {
                pos--;

                Prepara_Possessio(pos);
                Envia_MAX7219_2(DIG2, PosUnitat);
                Envia_MAX7219_2(DIG1, PosDesena);

                CPosUnitat[0]=(char)(48+PosUnitat);
                CPosDesena[0]=(char)(48+PosDesena);

                LCD_1_Position(1,7);
                LCD_1_PrString(CPosDesena);
                LCD_1_Position(1,8);
                LCD_1_PrString(CPosUnitat);
        }
        else
        {
                Timer16_1_Stop();
                Botzina2();
                aux1=0;
        }
        I2Cm_1_Start();
}

void Prepara_Temps(BYTE enter1)
{
        desena=enter1/10;
        unitat=enter1%10;
}

void Prepara_Possessio(BYTE enter2)
{
        PosDesena=enter2/10;
        PosUnitat=enter2%10;
}

void Botzina1()
{
        int j,k,l;

        for(l=0;l<20;l++)
        {
                for(k=0;k<1000;k++)
                {
                        Envia_MAX7219_3(DIG4, 0x02);
                }
                for(j=0;j<1000;j++);
        }
}

void Botzina2()
{
        int j,k,l;

        for(l=0;l<20;l++)
        {
                for(k=0;k<1000;k++)
                {
                        Envia_MAX7219_3(DIG4, 0x02);
                }
                for(j=0;j<2000;j++);
        }
```

```
}

void Envia_MAX7219_1(BYTE registre, BYTE dades)
{
/*
Cada bit de la línia DATA entra a un registre d'espera de 16 bits al ritme del senyal de rellotge.
Un flanc de pujada en el senyal LOAD fa que els bits entrin als registres.

DATA __________|D15|D14|D13|D12|D11|D10|D09|D08|D07|D06|D05|D04|D03|D02|D01|D00|______
     _________   __  __  __  __  __  __  __  __  __  __  __  __  __  __  _________
CLK          |__|  |__|  |__|  |__|  |__|  |__|  |__|  |__|  |__|  |__|  |__|
              ____________________________________________________________________ ___
LOAD ______|                                                                    |__|
*/

        LOAD_1;                         // Es provoca un flanc de pujada al senyal LOAD
        SPIM_1_SendTxData(NO_OP);       // No-Op Codes pels altres drivers
        SPIM_1_SendTxData(0x00);
        SPIM_1_SendTxData(NO_OP);
        SPIM_1_SendTxData(0x00);
        SPIM_1_SendTxData(registre);    // Registre al que es vol accedir
        SPIM_1_SendTxData(dades);       // Dades que s'envien al registre
        LOAD_0;                         // Flanc de baixada
        LOAD_1;                         // Flanc de pujada, els 16 bits son introduits als registres

        I2Cm_1_Start();
}

void Envia_MAX7219_2(BYTE registre, BYTE dades)
{
        LOAD_1;
        SPIM_1_SendTxData(NO_OP);
        SPIM_1_SendTxData(0x00);
        SPIM_1_SendTxData(registre);
        SPIM_1_SendTxData(dades);
        SPIM_1_SendTxData(NO_OP);
        SPIM_1_SendTxData(0x00);
        LOAD_0;
        LOAD_1;

        I2Cm_1_Start();
}

void Envia_MAX7219_3(BYTE registre, BYTE dades)
{
        LOAD_1;
        SPIM_1_SendTxData(registre);
        SPIM_1_SendTxData(dades);
        SPIM_1_SendTxData(NO_OP);
        SPIM_1_SendTxData(0x00);
        SPIM_1_SendTxData(NO_OP);
        SPIM_1_SendTxData(0x00);
        LOAD_0;
        LOAD_1;

        I2Cm_1_Start();
}

void Inicialitza_MAX7219()
{
/*
Inicialment, després d'alimentar el dispositiu, els registres són resetejats, els displays
s'apaguen i entra en mode SHUTDOWN. Aquest mode no impedeix configurar els registres. A
continuació es configuren segons les necessitats i al final es surt del mode SHUTDOWN (i també
del mode DISPLAY TEST, que també està activat predeterminadament).
*/
        unsigned int i;                 // Per arribar fins a i=65536 (2 elevat a 16)

        SPIM_1_Start(SPIM_1_SPIM_MODE_0 | SPIM_1_SPIM_MSB_FIRST);

        Envia_MAX7219_1(SCAN_LIMIT, 0x07);   // S'accedeix al SCAN LIMIT REGISTER, s'activen els
                                             // dígits 0-7
        Envia_MAX7219_1(INTENSITY, 0x0f);    // S'accedeix al INTENSITY REGISTER, duty cicle de
                                             // 31/32
        Envia_MAX7219_1(DECODE, 0xef);       // S'accedeix al DECODE MODE REGISTER, decodificació
                                             // BCD pels dígits 0-3 i 5-7
        Envia_MAX7219_1(DIG0, 0x01);         // S'accedeix al DÍGIT 0, s'hi introdueix un "1"
```

```
        Envia_MAX7219_1(DIG1, 0x00);          // S'accedeix al DÍGIT 1, s'hi introdueix un "0"
        Envia_MAX7219_1(DIG2, 0x00);          // S'accedeix al DÍGIT 2, s'hi introdueix un "0"
        Envia_MAX7219_1(DIG3, 0x00);          // S'accedeix al DÍGIT 3, s'hi introdueix un "0"
        Envia_MAX7219_1(DIG4, 0x00);          // S'accedeix al DÍGIT 4, estat apagat
        Envia_MAX7219_1(DIG5, 0x00);          // S'accedeix al DÍGIT 5, s'hi introdueix un "0"
        Envia_MAX7219_1(DIG6, 0x00);          // S'accedeix al DÍGIT 6, s'hi introdueix un "0"
        Envia_MAX7219_1(DIG7, 0x0f);          // S'accedeix al DÍGIT 7, estat apagat

        Envia_MAX7219_2(SCAN_LIMIT, 0x07);    // S'accedeix al SCAN LIMIT REGISTER, s'activen els
                                              // dígits 0-6
        Envia_MAX7219_2(INTENSITY, 0x0f);     // S'accedeix al INTENSITY REGISTER, duty cicle de
                                              // 31/32
        Envia_MAX7219_2(DECODE, 0x6f);        // S'accedeix al DECODE MODE REGISTER, decodificació
                                              // BCD pels dígits 0-3 i 5-6
        Envia_MAX7219_2(DIG0, 0x00);          // S'accedeix al DÍGIT 0, s'hi introdueix un "0"
        Envia_MAX7219_2(DIG1, 0x02);          // S'accedeix al DÍGIT 1, s'hi introdueix un "2"
        Envia_MAX7219_2(DIG2, 0x04);          // S'accedeix al DÍGIT 2, s'hi introdueix un "4"
        Envia_MAX7219_2(DIG3, 0x00);          // S'accedeix al DÍGIT 3, s'hi introdueix un "0"
        Envia_MAX7219_2(DIG4, 0x00);          // S'accedeix al DÍGIT 4, estat apagat
        Envia_MAX7219_2(DIG5, 0x00);          // S'accedeix al DÍGIT 5, s'hi introdueix un "0"
        Envia_MAX7219_2(DIG6, 0x00);          // S'accedeix al DÍGIT 6, s'hi introdueix un "0"

        Envia_MAX7219_3(SCAN_LIMIT, 0x04);    // S'accedeix al SCAN LIMIT REGISTER, s'activen els
                                              // dígits 0-4
        Envia_MAX7219_3(INTENSITY, 0x0f);     // S'accedeix al INTENSITY REGISTER, duty cicle de
                                              // 31/32
        Envia_MAX7219_3(DECODE, 0x00);        // S'accedeix al DECODE MODE REGISTER, no
                                              // decodificació BCD
        Envia_MAX7219_3(DIG0, 0x00);          // Leds i botzina apagats
        Envia_MAX7219_3(DIG1, 0x00);
        Envia_MAX7219_3(DIG2, 0x00);
        Envia_MAX7219_3(DIG3, 0x00);
        Envia_MAX7219_3(DIG4, 0x00);

        Envia_MAX7219_1(SHUTDOWN, 0x01);      // S'accedeix al SHUTDOWN REGISTER, s'escull el mode
                                              // "normal operation"
        Envia_MAX7219_2(SHUTDOWN, 0x01);
        Envia_MAX7219_3(SHUTDOWN, 0x01);

        Envia_MAX7219_1(DISPLAY_TEST, 0x01); // S'accedeix al DISPLAY TEST REGISTER, s'escull el
                                              // mode "Display Test Mode"
        Envia_MAX7219_2(DISPLAY_TEST, 0x01);
       Envia_MAX7219_3(DISPLAY_TEST, 0x01);
        for(i=0;i<50000;i++){                 // Bucle per encesa de tots els leds (i botzina) per
                                              // la seva comprovació
        }
        Envia_MAX7219_1(DISPLAY_TEST, 0x00); // S'accedeix al DISPLAY TEST REGISTER, s'escull el
                                              // mode "normal operation"
        Envia_MAX7219_2(DISPLAY_TEST, 0x00);
        Envia_MAX7219_3(DISPLAY_TEST, 0x00);
}

void Inicialitza_LCD()
{
        LCD_1_Start();                        // Funció API per activar USER MODULE LCD
        LCD_1_Position(0,0);                  // API per posició al LCD
        LCD_1_PrCString("LOC 10:00 0  VIS"); // API per imprimir al LCD
        LCD_1_Position(1,0);
        LCD_1_PrCString("000 0  24 0  000");
}

void PremTecla()
{
        if(PolsadorTMV)                                         // Temps mort local
        {
                TmortL++;
                if(TmortL==1)
                        Envia_MAX7219_3(DIG0, 0x10);
                else if(TmortL==2)
                        Envia_MAX7219_3(DIG0, 0x30);
                else if(TmortL==3)
                        Envia_MAX7219_3(DIG0, 0x70);
        }
        else if(PolsadorTML)                                    // Temps mort visitant
        {
                TmortV++;
                if(TmortV==1)
```

```
                Envia_MAX7219_3(DIG1, 0x04);
        else if(TmortV==2)
                Envia_MAX7219_3(DIG1, 0x0c);
        else if(TmortV==3)
                Envia_MAX7219_3(DIG1, 0x1c);
}
else if(PolsadorTMR)                                    // Reset temps morts
{
        TmortL=0;
        TmortV=0;
        Envia_MAX7219_3(DIG0, 0x00);
        Envia_MAX7219_3(DIG1, 0x00);
}
else if(PCFint==0)
{
        BYTE rxBuf1[], rxBuf2[], rxBuf3[];

        I2Cm_1_fReadBytes(0x20,rxBuf1,1,I2Cm_1_CompleteXfer);     // Expansor 1, adreça
                                                                  // 0x20=0100 000

        if(rxBuf1[0]==0xfe)                         // Expansor 1, pin 0 - RESET TEMPTEIG
        {
                TanteoLoc=0;
                TanteoVis=0;

                LCD_1_Position(1,0);
                LCD_1_PrCString("000");
                LCD_1_Position(1,13);
                LCD_1_PrCString("000");
                Envia_MAX7219_1(DIG4, 0x00);
                Envia_MAX7219_1(DIG5, 0x0f);
                Envia_MAX7219_1(DIG6, 0x0f);
                Envia_MAX7219_2(DIG4, 0x00);
                Envia_MAX7219_2(DIG5, 0x0f);
                Envia_MAX7219_2(DIG6, 0x0f);
        }
        else if(rxBuf1[0]==0xfd)            // Expansor 1, pin 1 - TEMPTEIG LOCAL +1
        {
                TanteoLoc++;

                BYTE uL, dL, cL;
                cL=TanteoLoc/100;
                dL=(TanteoLoc%100)/10;
                uL=(TanteoLoc%10)%10;

                if(cL==1)
                        Envia_MAX7219_1(DIG4, 0x30);

                Envia_MAX7219_1(DIG5, dL);
                Envia_MAX7219_1(DIG6, uL);

                char CcL[2], CdL[2],CuL[2];
                CcL[0]=(char)(48+cL);
                CdL[0]=(char)(48+dL);
                CuL[0]=(char)(48+uL);
                LCD_1_Position(1,0);
                LCD_1_PrString(CcL);
                LCD_1_Position(1,1);
                LCD_1_PrString(CdL);
                LCD_1_Position(1,2);
                LCD_1_PrString(CuL);
        }
        else if(rxBuf1[0]==0xfb)            // Expansor 1, pin 2 - TEMPTEIG LOCAL -1
        {
                if(TanteoLoc>0)
                        TanteoLoc--;

                BYTE uL, dL, cL;
                cL=TanteoLoc/100;
                dL=(TanteoLoc%100)/10;
                uL=(TanteoLoc%10)%10;

                if(cL==1)
                        Envia_MAX7219_1(DIG4, 0x30);
                else
                        Envia_MAX7219_1(DIG4, 0x00);
```

```
                Envia_MAX7219_1(DIG5, dL);
                Envia_MAX7219_1(DIG6, uL);

                char CcL[2], CdL[2],CuL[2];
                CcL[0]=(char)(48+cL);
                CdL[0]=(char)(48+dL);
                CuL[0]=(char)(48+uL);
                LCD_1_Position(1,0);
                LCD_1_PrString(CcL);
                LCD_1_Position(1,1);
                LCD_1_PrString(CdL);
                LCD_1_Position(1,2);
                LCD_1_PrString(CuL);
        }
        else if(rxBuf1[0]==0xf7)                    // Expansor 1, pin 3 - FALTES LOCALS +1
        {
                FaltesLoc++;
                if(FaltesLoc<=5)
                {
                        CFaltesLoc[0]=(char)(48+FaltesLoc);
                        LCD_1_Position(1,4);
                        LCD_1_PrString(CFaltesLoc);

                        Envia_MAX7219_2(DIG0, FaltesLoc);
                }
        }
        else if(rxBuf1[0]==0xef)                    // Expansor 1, pin 4 - POSSESSIÓ LOCAL
        {
                Envia_MAX7219_3(DIG0, 0x04);
        }
        else if(rxBuf1[0]==0xdf)                    // Expansor 1, pin 5 - RESET FALTES
        {
                FaltesLoc=0, FaltesVis=0;
                LCD_1_Position(1,4);
                LCD_1_PrCString("0");
                LCD_1_Position(1,10);
                LCD_1_PrCString("0");

                Envia_MAX7219_2(DIG0, FaltesLoc);
                Envia_MAX7219_2(DIG3, FaltesVis);
        }
        else if(rxBuf1[0]==0xbf)                    // Expansor 1, pin 6 - PERIODE
        {
                periode++;

                if(periode<=8)
                {
                        Cperiode[0]=(char)(48+periode);
                        if(periode!=0)
                                Envia_MAX7219_1(DIG7, periode);
                        else
                                Envia_MAX7219_1(DIG7, 0x0f);
                }
                else
                {
                        Cperiode[0]='E';
                        periode=-1;
                        Envia_MAX7219_1(DIG7, 0x0b);
                }

                LCD_1_Position(0,10);
                LCD_1_PrString(Cperiode);
        }
        else if(rxBuf1[0]==0x7f)            // Expansor 1, pin 7 - POSSESSIÓ VISITANT
        {
                Envia_MAX7219_3(DIG0, 0x08);
        }

        I2Cm_1_fReadBytes(0x21,rxBuf2,1,I2Cm_1_CompleteXfer);      // Expansor 2, adreça
                                                                   // 0x21=0100 001

        if(rxBuf2[0]==0xfe)                 // Expansor 2, pin 0 - FALTES VISITANTS +1
        {
                FaltesVis++;
                if(FaltesVis<=5)
                {
                        CFaltesVis[0]=(char)(48+FaltesVis);
```

```
                        LCD_1_Position(1,10);
                        LCD_1_PrString(CFaltesVis);

                        Envia_MAX7219_2(DIG3, FaltesVis);
                }
        }
        else if(rxBuf2[0]==0xfd)      // Expansor 2, pin 1 - CANVI DE SENTIT EN POSSESSIÓ
        {
                if(AuxSentit==0)
                {
                        Envia_MAX7219_3(DIG3, 0x00);
                        Envia_MAX7219_3(DIG2, 0x3c);
                        Envia_MAX7219_3(DIG1, 0x60);
                        AuxSentit=1;
                }
                else
                {
                        Envia_MAX7219_3(DIG1, 0x00);
                        Envia_MAX7219_3(DIG2, 0x40);
                        Envia_MAX7219_3(DIG3, 0x7c);
                        AuxSentit=0;
                }
        }
        else if(rxBuf2[0]==0xfb)             // Expansor 2, pin 2 - TEMPTEIG VISITANT +1
        {
                TanteoVis++;

                BYTE uV, dV, cV;
                cV=TanteoVis/100;
                dV=(TanteoVis%100)/10;
                uV=(TanteoVis%10)%10;

                if(cV==1)
                        Envia_MAX7219_2(DIG4, 0x30);

                Envia_MAX7219_2(DIG5, dV);
                Envia_MAX7219_2(DIG6, uV);

                char CcV[2], CdV[2],CuV[2];
                CcV[0]=(char)(48+cV);
                CdV[0]=(char)(48+dV);
                CuV[0]=(char)(48+uV);
                LCD_1_Position(1,13);
                LCD_1_PrString(CcV);
                LCD_1_Position(1,14);
                LCD_1_PrString(CdV);
                LCD_1_Position(1,15);
                LCD_1_PrString(CuV);
        }
        else if(rxBuf2[0]==0xf7)             // Expansor 2, pin 3 - TEMPTEIG VISITANT -1
        {
                if(TanteoVis>0)
                        TanteoVis--;

                BYTE uV, dV, cV;
                cV=TanteoVis/100;
                dV=(TanteoVis%100)/10;
                uV=(TanteoVis%10)%10;

                if(cV==1)
                        Envia_MAX7219_2(DIG4, 0x30);
                else
                        Envia_MAX7219_2(DIG4, 0x00);

                Envia_MAX7219_2(DIG5, dV);
                Envia_MAX7219_2(DIG6, uV);

                char CcV[2], CdV[2],CuV[2];
                CcV[0]=(char)(48+cV);
                CdV[0]=(char)(48+dV);
                CuV[0]=(char)(48+uV);
                LCD_1_Position(1,13);
                LCD_1_PrString(CcV);
                LCD_1_Position(1,14);
                LCD_1_PrString(CdV);
                LCD_1_Position(1,15);
                LCD_1_PrString(CuV);
```

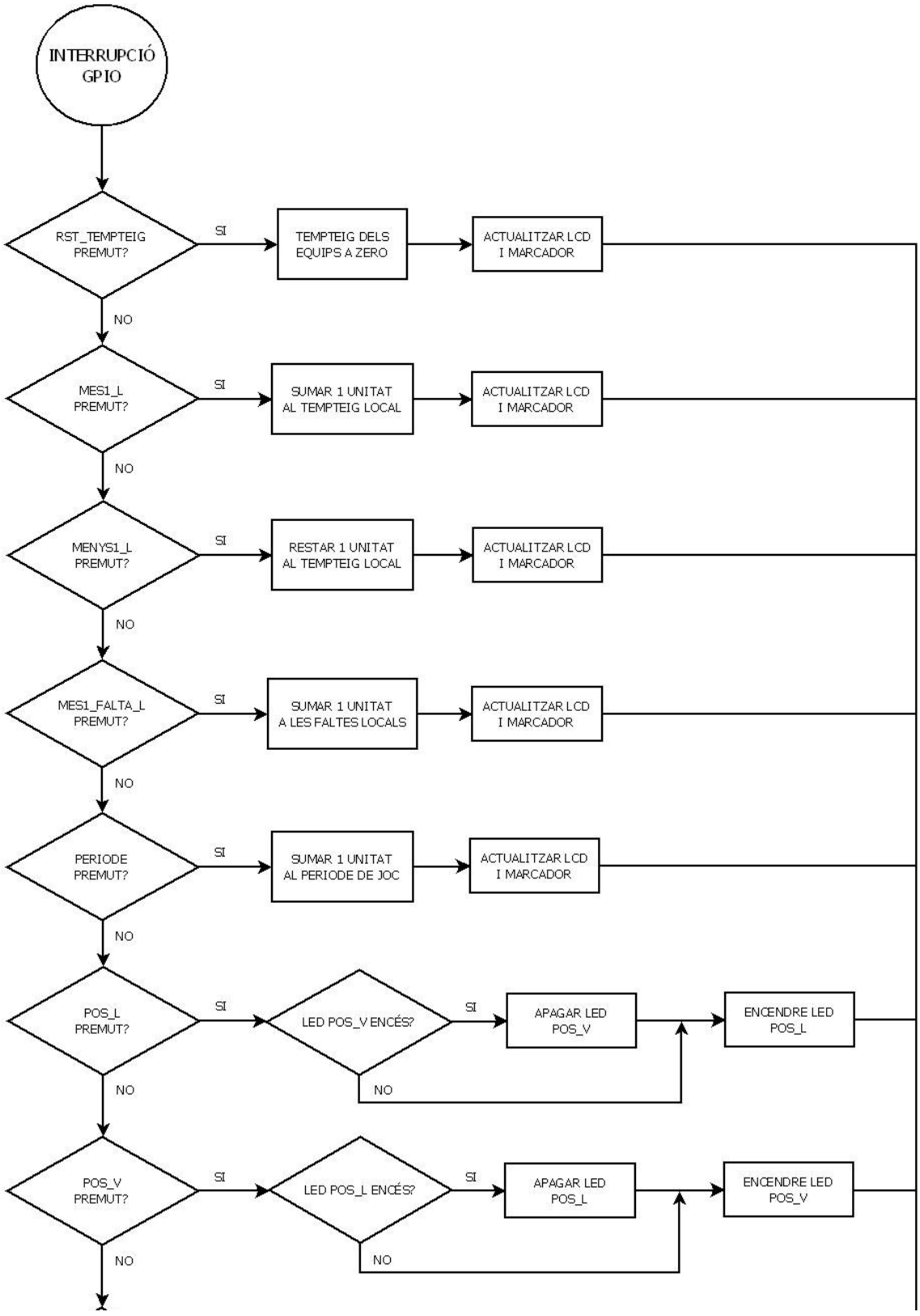
```
                }
                else if(rxBuf2[0]==0xef)                    // Expansor 2, pin 4 - START/STOP CRONO
                {
                        if(aux2==0)                         // Start/stop temps
                        {
                                aux2++;
                                Timer16_2_Start();
                        }
                        else
                        {
                                Timer16_2_Stop();
                                aux2--;
                        }
                }
                else if(rxBuf2[0]==0xdf)                    // Expansor 2, pin 5 - RESET CRONO
                {
                        minut=10, segon=0, centesima=0;

                        Envia_MAX7219_1(DIG3, 0x00);
                        Envia_MAX7219_1(DIG2, 0x00);
                        Envia_MAX7219_1(DIG1, 0x00);
                        Envia_MAX7219_1(DIG0, 0x01);

                        LCD_1_Position(0,4);
                        LCD_1_PrCString("10:00");

                        I2Cm_1_Start();
                }
                else if(rxBuf2[0]==0xbf)                    // Expansor 2, pin 6 - CRONO -1 MINUT
                {
                        if(minut>0)
                        {
                                minut--;

                                Prepara_Temps(minut);
                                Envia_MAX7219_1(DIG1, unitat);
                                Envia_MAX7219_1(DIG0, desena);

                                Cunitat[0]=(char)(48+unitat);
                                Cdesena[0]=(char)(48+desena);

                                LCD_1_Position(0,4);
                                LCD_1_PrString(Cdesena);
                                LCD_1_Position(0,5);
                                LCD_1_PrString(Cunitat);
                        }
                }
                else if(rxBuf2[0]==0x7f)                    // Expansor 2, pin 7 - CRONO +1 MINUT
                {
                        minut++;

                        Prepara_Temps(minut);
                        Envia_MAX7219_1(DIG1, unitat);
                        Envia_MAX7219_1(DIG0, desena);

                        Cunitat[0]=(char)(48+unitat);
                        Cdesena[0]=(char)(48+desena);

                        LCD_1_Position(0,4);
                        LCD_1_PrString(Cdesena);
                        LCD_1_Position(0,5);
                        LCD_1_PrString(Cunitat);
                }

                I2Cm_1_fReadBytes(0x23,rxBuf3,1,I2Cm_1_CompleteXfer);     // Expansor 3, adreça
                                                                          // 0x23=0100 011

                if(rxBuf3[0]==0xfe)                         // Expansor 3, pin 0 - SENSE CONNEXIÓ
                {

                }
                else if(rxBuf3[0]==0xfd)                    // Expansor 3, pin 1 - CRONO +1 SEGON
                {
                        segon++;

                        Prepara_Temps(segon);
                        Envia_MAX7219_1(DIG3, unitat);
```

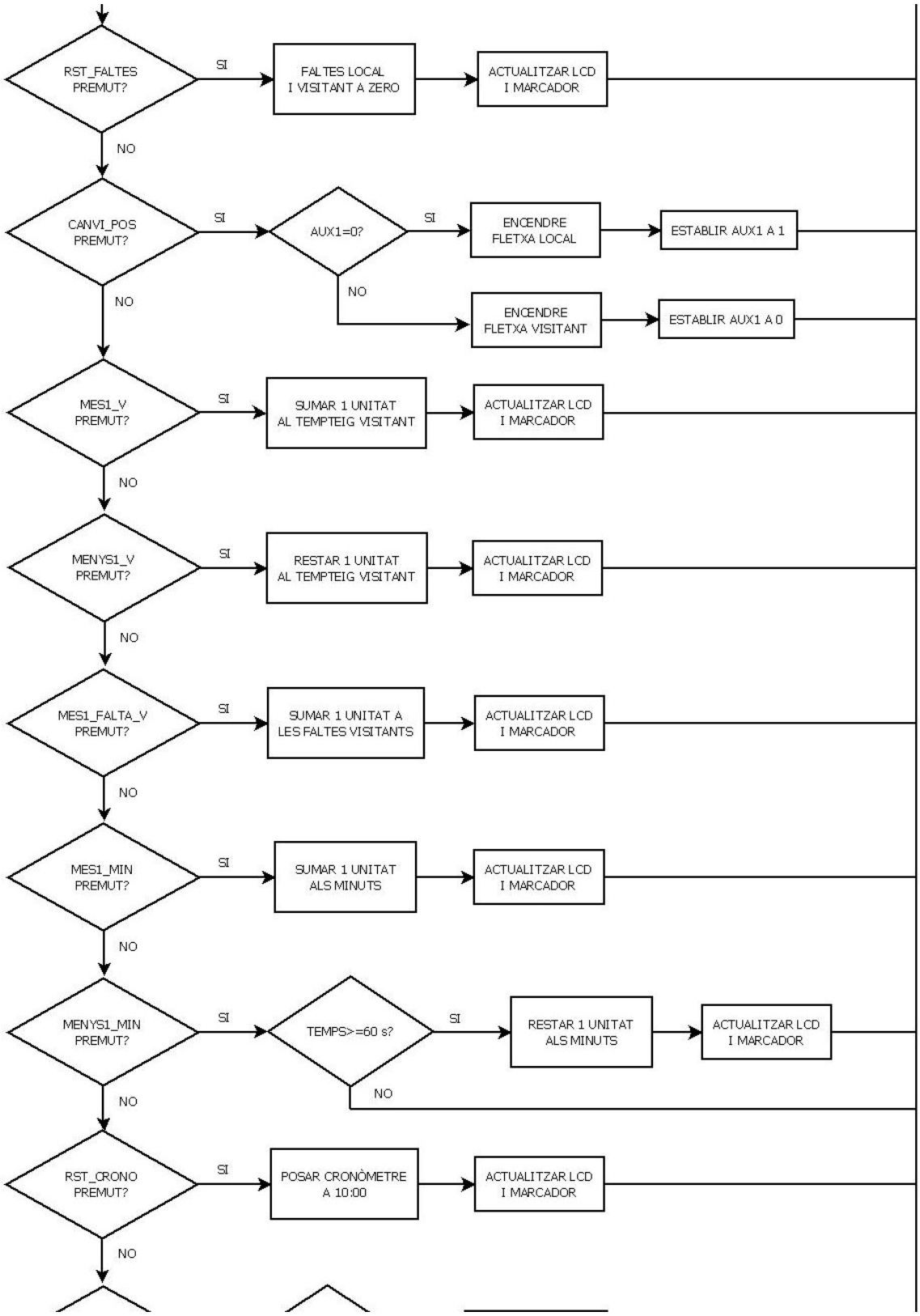
```
                Envia_MAX7219_1(DIG2, desena);

                Cunitat[0]=(char)(48+unitat);
                Cdesena[0]=(char)(48+desena);

                LCD_1_Position(0,7);
                LCD_1_PrString(Cdesena);
                LCD_1_Position(0,8);
                LCD_1_PrString(Cunitat);
        }
        else if(rxBuf3[0]==0xfb)                    // Expansor 3, pin 2 - CRONO -1 SEGON
        {
                if(minut>0&&segon>0)
                {
                        segon--;

                        Prepara_Temps(segon);
                        Envia_MAX7219_1(DIG3, unitat);
                        Envia_MAX7219_1(DIG2, desena);

                        Cunitat[0]=(char)(48+unitat);
                        Cdesena[0]=(char)(48+desena);

                        LCD_1_Position(0,7);
                        LCD_1_PrString(Cdesena);
                        LCD_1_Position(0,8);
                        LCD_1_PrString(Cunitat);
                }
        }
        else if(rxBuf3[0]==0xf7)                    // Expansor 3, pin 3 - BOTZINA
        {
                Botzina1();
        }
        else if(rxBuf3[0]==0xef)                    // Expansor 3, pin 4 - RESET POSSESSIÓ
        {
                pos=24;                         // Reset possessió a 24
                Prepara_Possessio(pos);

                Envia_MAX7219_2(DIG2, PosUnitat);
                Envia_MAX7219_2(DIG1, PosDesena);

                CPosUnitat[0]=(char)(48+PosUnitat);
                CPosDesena[0]=(char)(48+PosDesena);

                LCD_1_Position(1,7);
                LCD_1_PrString(CPosDesena);
                LCD_1_Position(1,8);
                LCD_1_PrString(CPosUnitat);

                I2Cm_1_Start();
        }
        else if(rxBuf3[0]==0xdf)            // Expansor 3, pin 5 - POSSESSIÓ -1 SEGON
        {
                if(pos>1)
                {
                        pos--;

                        Prepara_Possessio(pos);

                        Envia_MAX7219_2(DIG2, PosUnitat);
                        Envia_MAX7219_2(DIG1, PosDesena);

                        CPosUnitat[0]=(char)(48+PosUnitat);
                        CPosDesena[0]=(char)(48+PosDesena);

                        LCD_1_Position(1,7);
                        LCD_1_PrString(CPosDesena);
                        LCD_1_Position(1,8);
                        LCD_1_PrString(CPosUnitat);
                }
        }
        else if(rxBuf3[0]==0xbf)            // Expansor 3, pin 6 - POSSESSIÓ +1 SEGON
        {
                if(pos<24)
                {
                        pos++;
```

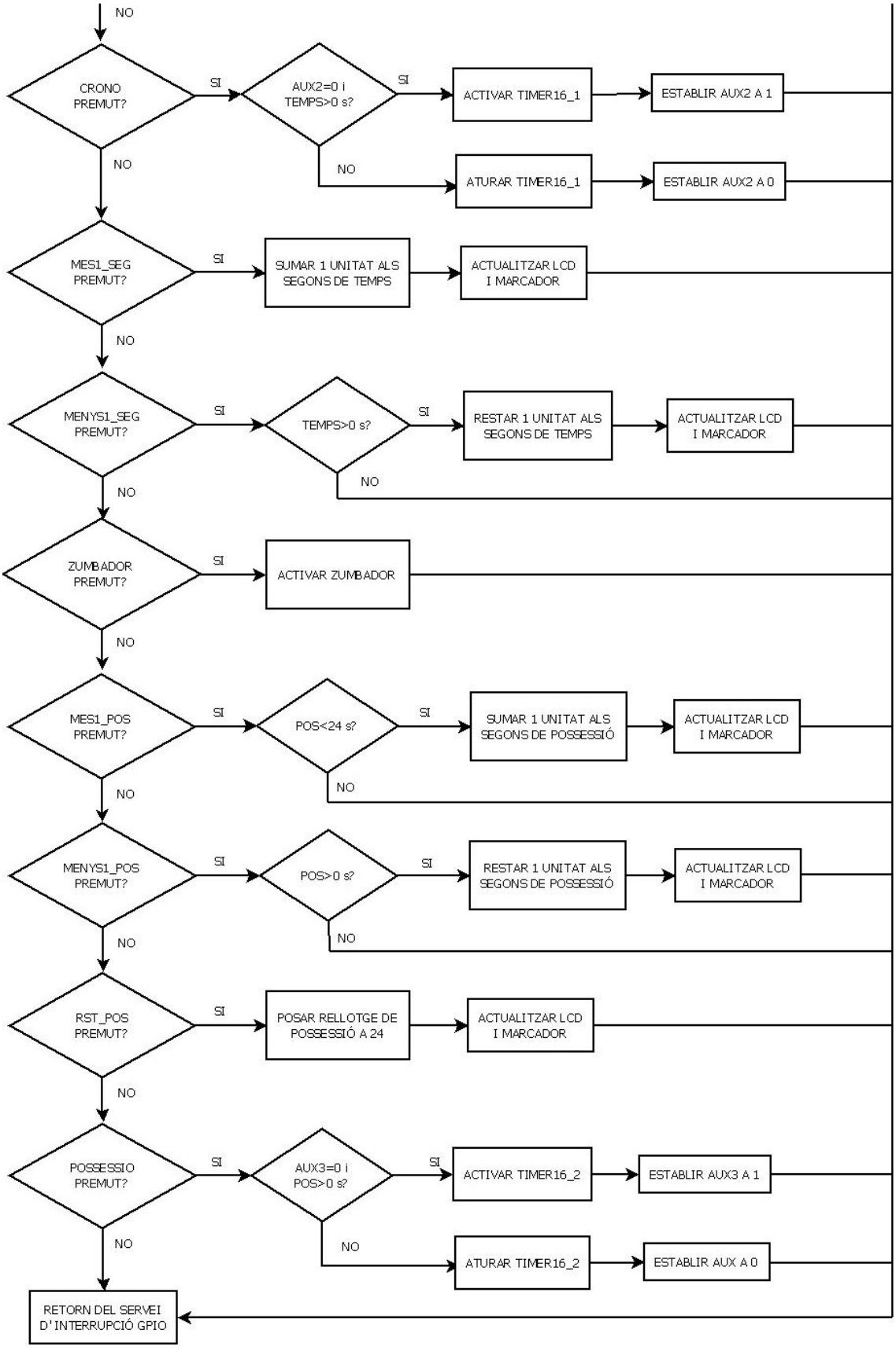
```
                                Prepara_Possessio(pos);

                                Envia_MAX7219_2(DIG2, PosUnitat);
                                Envia_MAX7219_2(DIG1, PosDesena);

                                CPosUnitat[0]=(char)(48+PosUnitat);
                                CPosDesena[0]=(char)(48+PosDesena);

                                LCD_1_Position(1,7);
                                LCD_1_PrString(CPosDesena);
                                LCD_1_Position(1,8);
                                LCD_1_PrString(CPosUnitat);
                        }
                }
                else if(rxBuf3[0]==0x7f)            // Expansor 3, pin 7 - START/STOP POSSESSIÓ
                {
                        if(aux1==0)
                        {
                                aux1=1;
                                Timer16_1_Start();
                        }
                        else
                        {
                                Timer16_1_Stop();
                                aux1=0;
                        }
                }
        }
}

void main()
{
        Inicialitza_LCD();
        Inicialitza_MAX7219();

        M8C_EnableGInt;                             // Habilitació de les interrupcions generals
        M8C_EnableIntMask(INT_MSK0,INT_MSK0_GPIO);
        Timer16_2_EnableInt();                      // Habilitació de la interrupció del Timer8_1
        Timer16_1_EnableInt();                      // Habilitació de la interrupció del Timer16_1

        I2Cm_1_Start();                             // Inici de la comunicació amb els expansors
}
```

4.4.3 Annex 3: diagrama de flux

Aquest és el diagrama de flux de les interrupcions GPIO del programa final:




RST_FALTES PREMUT?
SI
FALTES LOCAL I VISITANT A ZERO
ACTUALITZAR LCD I MARCADOR
NO
CANVI_POS PREMUT?
SI
AUX1=0?
SI
ENCENDRE FLETXA LOCAL
ESTABLIR AUX1 A 1
NO
ENCENDRE FLETXA VISITANT
ESTABLIR AUX1 A 0
NO
MES1_V PREMUT?
SI
SUMAR 1 UNITAT AL TEMPTEIG VISITANT
ACTUALITZAR LCD I MARCADOR
NO
MENYS1_V PREMUT?
SI
RESTAR 1 UNITAT AL TEMPTEIG VISITANT
ACTUALITZAR LCD I MARCADOR
NO
MES1_FALTA_V PREMUT?
SI
SUMAR 1 UNITAT A LES FALTES VISITANTS
ACTUALITZAR LCD I MARCADOR
NO
MES1_MIN PREMUT?
SI
SUMAR 1 UNITAT ALS MINUTS
ACTUALITZAR LCD I MARCADOR
NO
MENYS1_MIN PREMUT?
SI
TEMPS>=60 s?
SI
RESTAR 1 UNITAT ALS MINUTS
ACTUALITZAR LCD I MARCADOR
NO
NO
RST_CRONO PREMUT?
SI
POSAR CRONÒMETRE A 10:00
ACTUALITZAR LCD I MARCADOR
NO


NO
CRONO PREMUT?
SI
AUX2=0 i TEMPS>0 s?
SI
ACTIVAR TIMER16_1
ESTABLIR AUX2 A 1
NO
ATURAR TIMER16_1
ESTABLIR AUX2 A 0
NO
MES1_SEG PREMUT?
SI
SUMAR 1 UNITAT ALS SEGONS DE TEMPS
ACTUALITZAR LCD I MARCADOR
NO
MENYS1_SEG PREMUT?
SI
TEMPS>0 s?
SI
RESTAR 1 UNITAT ALS SEGONS DE TEMPS
ACTUALITZAR LCD I MARCADOR
NO
NO
ZUMBADOR PREMUT?
SI
ACTIVAR ZUMBADOR
NO
MES1_POS PREMUT?
SI
POS<24 s?
SI
SUMAR 1 UNITAT ALS SEGONS DE POSSESSIÓ
ACTUALITZAR LCD I MARCADOR
NO
NO
MENYS1_POS PREMUT?
SI
POS>0 s?
SI
RESTAR 1 UNITAT ALS SEGONS DE POSSESSIÓ
ACTUALITZAR LCD I MARCADOR
NO
NO
RST_POS PREMUT?
SI
POSAR RELLOTGE DE POSSESSIÓ A 24
ACTUALITZAR LCD I MARCADOR
NO
POSSESSIO PREMUT?
SI
AUX3=0 i POS>0 s?
SI
ACTIVAR TIMER16_2
ESTABLIR AUX3 A 1
NO
ATURAR TIMER16_2
ESTABLIR AUX A 0
NO
RETORN DEL SERVEI D'INTERRUPCIÓ GPIO